LEVEL (12)

AD

AMSAA

AMSAA

TECHNICAL REPORT NO. 225

AD A091355

SEEFAR:
AN IMPROVED MODEL FOR PRODUCING LINE-OF-SIGHT MAPS

BARBARA D. BROOME

SEPTEMBER 1980

APPROVED FOR PUBLIC RELEASE, DISTRIBUTION UNLIMITED.

U. S. ARMY MATERIEL SYSTEMS ANALYSIS ACTIVITY
ABERDEEN PROVING GROUND, MARYLAND

DISPOSITION

Destroy this report when no longer needed. Do not return it to the originator.

DISCLAIMER

The findings in this report are not to be construed as an official Department of the Army position unless so specified by other official documentation.

WARNING

Information and data contained in this document are based on the input available at the time of preparation. The results may be subject to change and should not be construed as representing the DARCOM position unless so specified.

TRADE NAMES

The use of trade names in this report does not constitute an official endorsement or approval of the use of such commercial hardware or software. The report may not be cited for purposes of advertisement.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: Technical Report No. 225

2. GOVT ACCESSION NO.: AD-A091 855

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle): SEEFAR: AN IMPROVED MODEL FOR PRODUCING LINE-OF-SIGHT MAPS

5. TYPE OF REPORT & PERIOD COVERED: Technical rept.

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s): Barbara D. Broome

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS: US Army Materiel Systems Analysis Activity, Aberdeen Proving Ground, Maryland

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: DA Project No. 1R665706M541

11. CONTROLLING OFFICE NAME AND ADDRESS: Cdr, US Army Materiel Development & Readiness Command, 5001 Eisenhower Avenue, Alexandria, VA 22333

12. REPORT DATE: September 1980

13. NUMBER OF PAGES: 86

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): 12 88

15. SECURITY CLASS. (of this report): Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

DTIC ELECTE NOV 21 1980 S

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

intervisibility
line of sight
line-of-sight map

20. ABSTRACT

This report describes an improved model for producing line-of-sight maps. Many models determine whether a target is within view by drawing a terrain profile between observer and target and examining it to see if it interferes with line of sight; this requires generating a completely new profile for each target position. The new model avoids this time-consuming profile generation by dynamically recording the characteristics of a "running horizon" as computations are made for points further away from the observer. For each target point, a check is made to determine whether the target is behind the "horizon". This new approach results in a dramatic savings in both storage and computing time requirements.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

403711

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TABLE OF CONTENTS

LIST OF FIGURES

The next page is blank

ACKNOWLEDGEMENTS

The Author wishes to express special thanks to Paul Broome, Arthur Groves and Richard Kaste for their valuable contributions to the development of the SEEFAR algorithm, its implementation and the writing of this report.

The next page is blank.

SEEFAR: AN IMPROVED MODEL FOR PRODUCING LINE-OF-OF-SIGHT-MAPS

1. INTRODUCTION

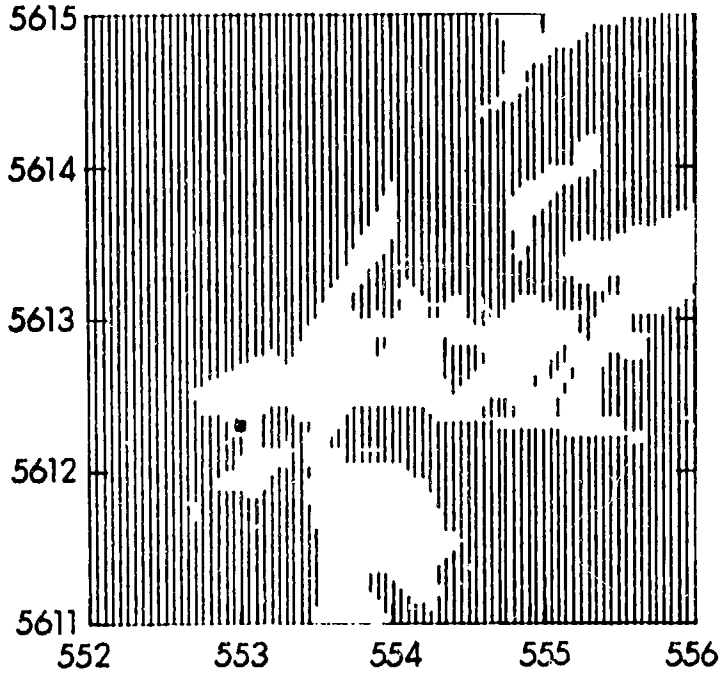
A line-of-sight map, or intervisibility plot, is a graphical representation of those portions of a geographical area which are visible to a given observer. (The term "visible" throughout this report refers to the existence of line of sight from an observation point to a target point which is uninterrupted by intervening terrain or obstacles.) These maps may be as shown in Figure 1, where the hatched portions represent those areas in which a target of a given height would be obscured. Targets in the blank portions would be visible. Or the maps may be shown as in Figure 2, where multiple target heights are considered. Each symbol represents a particular target height. A symbol plotted in a certain area indicates that targets of the associated height (or higher) located in this area are visible to the observer. For example, in Figure 2 a target 10 meters high (or higher) would be visible to the indicated observer if located at the point with UTM (Universal Transverse Mercalor Projection) coordinates (554500, 5612600).

A number of computer programs are available for providing these maps; most of these programs are based on the algorithm described as "an intuitive approach" in this report. LOSMAP is one such program available at USAMSAA. Use of an alternate algorithm, however, has resulted in a considerable reduction in both the memory requirements and compute time. An implementation of this algorithm, called SEEFAR, is included in Appendix A. The purpose of this report is to describe the basic SEEFAR algorithm and to compare it to the LOSMAP approach.

Line-of-sight maps have proved particularly useful in the development of combat scenarios for selecting reasonable observer positions and tactically sound attack routes. These maps can be useful to the wargamer, to the tactician in the field, and to the test plan developer. Additionally, they may be helpful in developing desirable weapon system characteristics. For example, it would be fruitless to develop a weapon system requiring line of sight with a range of 10 kilometers to be used in an area where observers typically cannot see a target at further than 5 kilometers.

The information required to produce these maps is of three types. First, the positional relationship between the map and observer must be specified. An area of interest is denoted by its extreme rectangular coordinates. The observer's position must be within this area. Secondly, heights of the observer and the target are required. The target height is the height of that point on the target that must be seen before the target is considered visible; the observer height designates the eye or sensor position above the ground. Finally, the actual terrain elevations along with any ancillary terrain data must be supplied.

The Defense Mapping Agency (DMA) digitized terrain data fulfill the third information requirement. These data were created by extracting elevations from contour lines on 1:50000 scale contour maps. Then, through planar interpolation, elevations were obtained at 12.5-meter intervals. This information

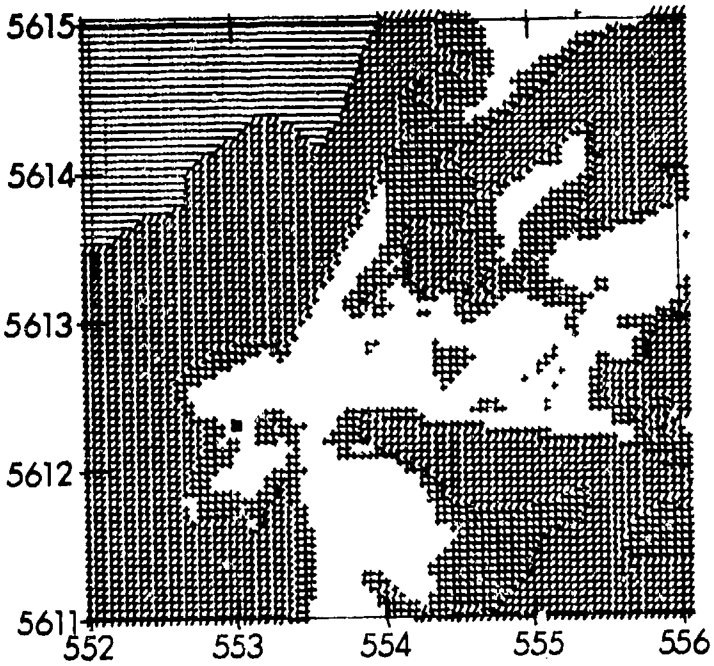


OBSERVER COORDINATES (553000 , 5612300)

OBSERVER HEIGHT = 2.00

OBSERVER ID = 1

Figure 1. Line-of-Sight Map.
Single Target Height



OBSERVER COORDINATES (553000,5612300)
OBSERVER HEIGHT = 2.00
OBSERVER ID = 1

LEGEND
POSITION WHERE TARGET
X METERS HIGH OR MORE
CAN BE SEEN, WHERE

SYMBOL	X =
	2.00
+	10.00
/	100.00
-	NO TARGET <= 100.00 METERS HIGH CAN BE SEEN

Figure 2. Line-of-Sight Map
Multiple Target Height

is supplied on magnetic tapes as south-to-north strings of data sweeping from west to east. The term scanline, or scan, throughout this paper will denote a south-to-north string of information. Thus, the DMA data is a set of elevation scanlines. Additionally, an indication of the presence of a forest, orchard or urban area has been included with each elevation.

2. AN INTUITIVE LINE-OF-SIGHT ALGORITHM (LOSMAP)

2.1 Algorithm Description.

Now, given the information described in the previous section, suppose we wish to produce a map like that in Figure 1. Basically, at evenly spaced intervals thoughout the area of interest the question must be asked, "Can the observer see the target if it is at this spot?" A reasonable approach to answering this question is to draw the terrain profile between the observer and the target, then draw the line from the observer eyeball to the target point and determine whether the terrain profile interrupts that observer-target line. If there is no interruption, line of sight exists; otherwise, this spot is not visible.

In Figure 3 the observer and target are depicted in the array of digitized elevations. As the figure indicates, at each point where the line between observer and target crosses a grid line, linear interpolation is used to find another height on the desired profile. As the profile is thus being constructed, a comparison is made between the elevation of each new point on the profile and the corresponding height of the line drawn from the observer eyeball to the target point. If the profile height is ever the larger of the two heights, line of sight does not exist; otherwise, line of sight does exit.

Of course this is a simplification of the problem. Because the presence of vegetation and urban areas can have a major impact on visibility for an observer, it is generally desirable to use the DMA vegetation/urban indicators. For example, while constructing the profile an appropriate height increment may be added to the profile if it runs through trees or towns. Further, the effect of earth curvature must be considered. If multiple target heights are considered, some accounting method must be used to keep track of the minimum visible target height. This brief description gives enough information to point out the major problems with the algorithm: computer space and time requirements.

2.2 Problems Encountered: Storage and Compute Time

As long as the area of interest is small, there is no problem. The use of high speed computers enables one to perform the required calculations with ease.

Notice, however, that this algorithm requires easy access to elevations in the entire area of interest. For each target point, an observer-to-target profile is constructed, requiring the elevations of many intervening points. Consider making a 6 x 6 kilometer map. If a 12.5-meter grid is used, an array of elevations dimensioned 481 x 481 is needed. Storing one elevation per word would require 231,361 words of memory for the elevation array alone! This problem can be somewhat alleviated by using a thinned grid of 25, 50 or even 100 meters. While this thinning may result in more inaccuracy, the outcome is often deemed adequate. Further, these elevations may be packed into the elevation array several at a time thereby decreasing the required storage even more. But, inevitably, as the size of the area that can be handled is increased, a problem is encountered that requires an area slightly larger. Some have solved the

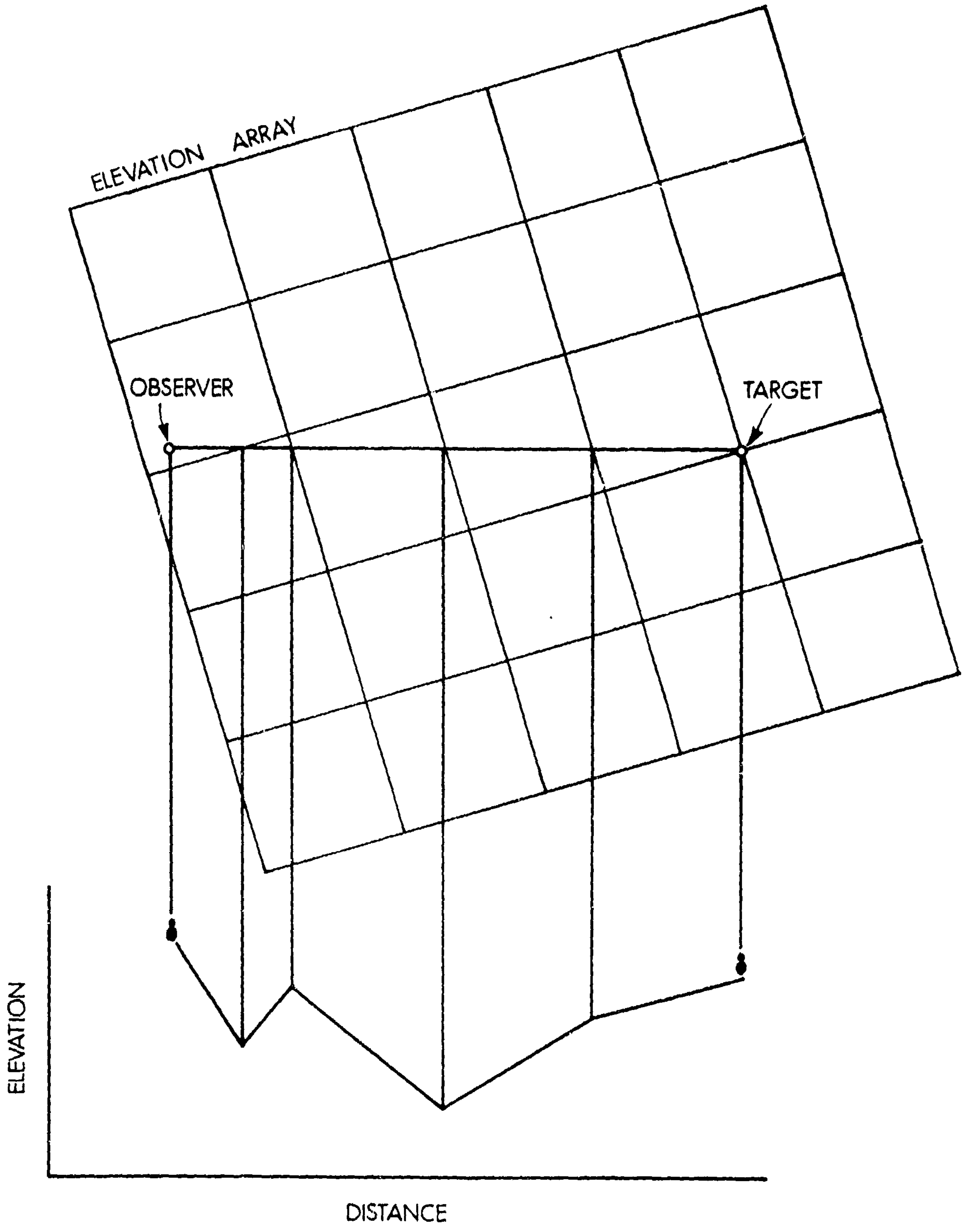


Figure 3. Constructing the Profile from Observer to Target.

storage problem by swapping data in and out of memory as needed. This, however, is rather cumbersome and will tremendously increase input-output time requirements. As one becomes concerned with larger areas of interest, as in air-to-ground and air-to-air intervisibility studies, this storage problem must be faced some other way.

Another problem to consider is that of compute time. If one needs a simple line-of-sight map like that in Figure 1, it is hard to predict exactly how many calculations will be performed. For each point on the map a profile must be started, so for an n x n map, at least n^2 calculations will be performed. If the profile interrupt is encountered early, few extra calculations will be needed. But, when line of sight to the target point exists, the entire profile must be constructed and each height comparison must be performed. With maps like that in Figure 2, considering multiple target heights, it becomes more likely that the entire profile must be constructed for every target point. The required number of calculations would then be of order n^3 (See Appendix C1). For large areas, this could easily be a prohibitively expensive algorithm.

3. DYNAMIC PROGRAMMING ALGORITHM (SEEFAR)

3.1 Algorithm Description.

A. One Ray. Now consider an alternate approach to the development of line-of-sight maps. How should line-of-sight calculations be performed if all the target positions lay along one line? For each target position along the ray the question must be answered, "How high must the target be raised to be seen?" But it hardly seems necessary to draw a new profile for each target position. Instead, one might choose to work from the observer out, extending the profile to each new target position, not keeping track of all the previously encountered elevations but simply updating the effect of the maximum interrupt encountered thus far as new interrupts are encountered. This maximum interrupt effect projected to the current target position is essentially the minimum target height required to overcome the effect of terrain between the observer and the target. This height can be thought of as a running horizon, running in the sense that it must be updated as target positions move further from the observer.

For each target position, then, the following procedure could be followed:

(1) Check to see if the current target position is above or below the current horizon (i.e., in or out of view)

(2) Plot the results

(3) Update the running horizon to reflect its movement outward, and any greater interrupts encountered. (This new horizon value will be used in the line of sight calcuation for the next target position.)

Consider now this running horizon, or maximum interrupt effect. Figure 4 demonstrates maximum interrupts, how they can change as the profile is extended, and how their corresponding projected effect must continually be updated. The maximum interrupt effect must be updated both as new interrupts are encountered and as the target position is changed. A comparison of Figures 4A and 4B demonstrates the need to update the running horizon as greater interrupts are encountered. Considering Figures 4B and 4C, it can be seen that the effect of the maximum interrupt will vary for different target positions even though the maximum itself may remain the same. One should further note that, as shown in Figure 5, the maximum interrupt is not necessarily the terrain profile with the highest elevation.

This procedure has several nice features. First, because it is not necessary to draw a new profile for each target position, fewer steps are required. Further, it is not necessary to store all the elevations between the observer and the target to determine whether line of sight exists. The elevation of the current target position and the horizon effect are the only values needed. This method can be extended to produce a full map, rather than a single ray, greatly reducing the storage and compute time problems.

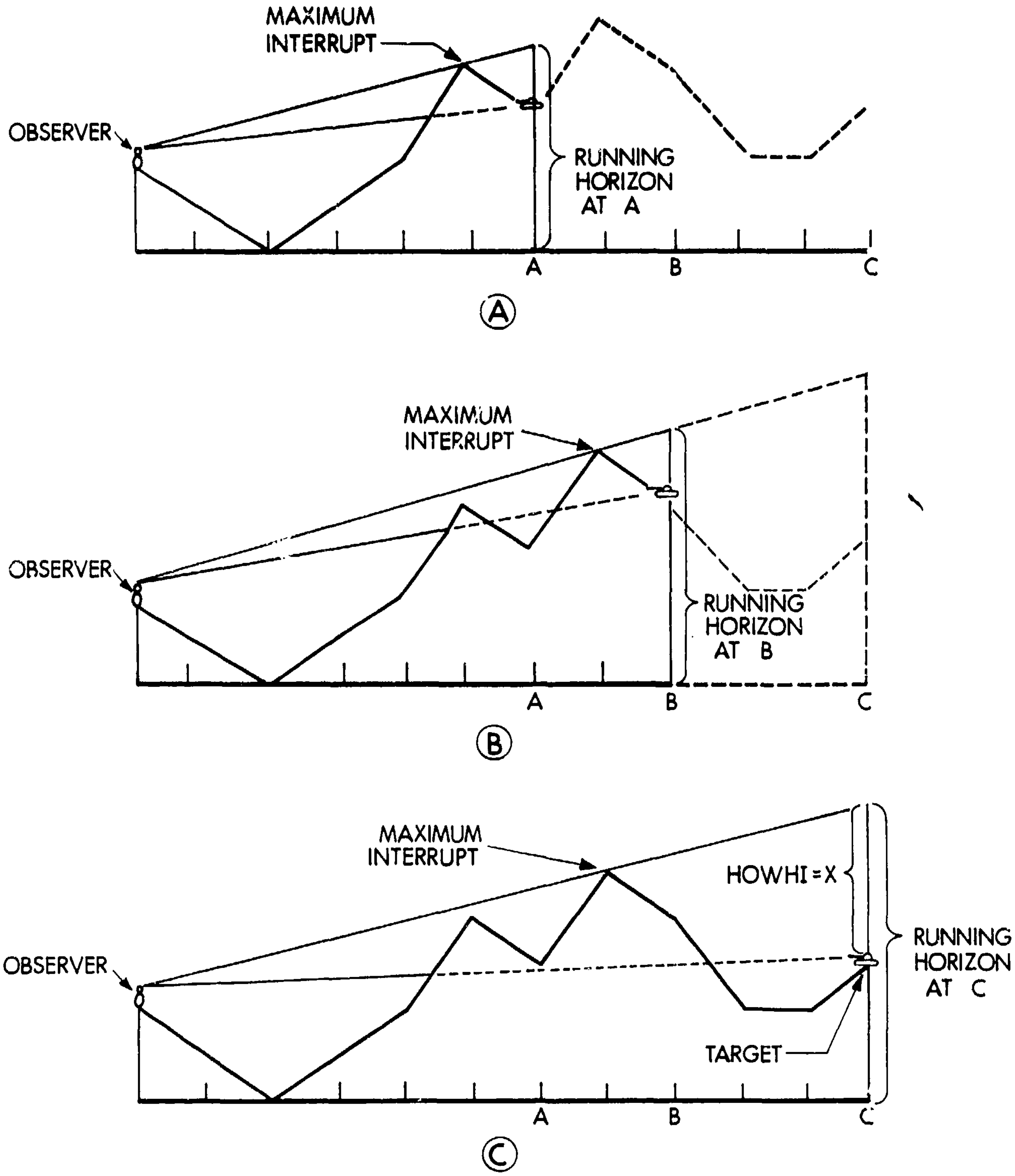


Figure 4. Updating the Running Horizon (The Effect of the Maximum Interrupt Projected to the Target Position)

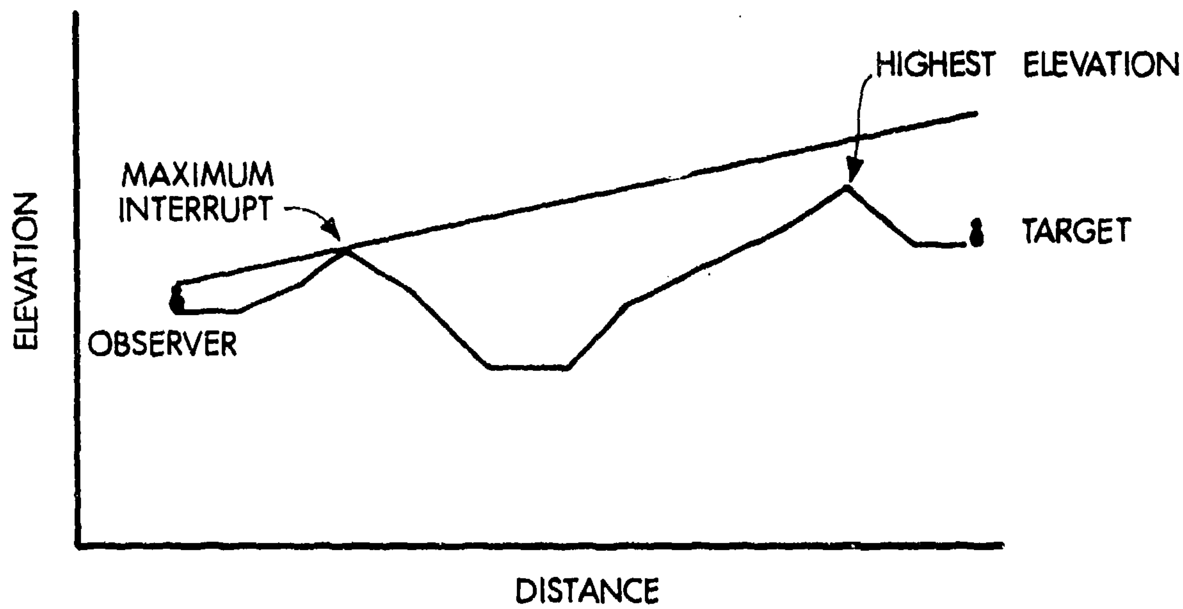


Figure 5. Maximum Interrupt Not Always Highest Elevation.

B. Full Map.

(1) Radial Map. Now how can this one-dimensional algorithm be expanded to two dimensions? One approach would be to produce a radial line-of-sight map, extending many profiles out from the observer. Certainly if line-of-sight information can be produced for one ray as described in the previous section, it would be a simple matter to project rays from the observer at small equally-space angle increments, combine the results and produce a two-dimensional map. With this method, however, the line-of-sight information would be very closely spaced near the observer and more widely spaced farther from the observer. As the profiles got further apart the likelihood of missing an important terrain feature would increase. Assuming, then, that evenly spaced data is more desirable, another method must be devised to produce the two-dimensional maps.

(2) Rectangular Map. As in the ray method the approach will be to work from the observer out (i.e., from the observer eastward, then from the observer westward), keeping track of a running horizon. But this time instead of having a running horizon point, there will be a south-to-north string of running horizon points (a horizon scanline) sweeping outward from the observer as the calculations are performed.

Remember, the map is actually a lattice of target spots. The goal is to determine for each target spot the value that denotes how high a target must be to be seen. Let HOWHI(i) be the height the ith target must be raised to be seen. If the target height is less than HOWHI(i) it cannot be seen; otherwise it can. The following sections describe a method for obtaining describe a method for obtaining these HOWHI(i)'s. In order to reach this goal one must continually keep up with HORIZON(i), the effect of the maximum interrupt encountered between the observer and the current target position.

Scans Closest to Observer

Because the plan is to work outward from the observer, the first step is to perform calculations for the two target scanlines closest to the observer. In checking for line of sight to the points on these scans, the first iterations of the east-running horizon and west-running horizon must also be formed. Figure 6 will help illustrate this procedure.

Consider the point 1R (as in Figure 6A). First determine how high a target must be to be seen at position 1R. There are obviously no terrain profile elevations between the observer and this point, so one might automatically say HOWHI(1R) = 0. But if forests and urban areas are to be considered a check must be made to determine whether such an obstacle is located at 1R. If so, HOWHI(1R) = 0 + HVEG(1R) where HVEG is the height of the obstacle. If the target height is less than HOWHI(1R), a target at 1R cannot be seen and the appropriate symbol must be plotted. But more must be done at this position. The running horizon value at 1R must be determined. The highest elevation (vegetation and urban included) encountered between the observer and 1R is simply the elevation at 1R (there are no elevations given elsewhere) plus the obstacle height at 1R. Thus, HORIZON (1R) = E(1R) + HVEG(1R) where E(1R) is the terrain elevation at 1R. A similar procedure for position 1L results in HOWHI(1L) = 0 + HVEG(1L) and HORIZON(1L) = E (1L) + HVEG(1L).

Now consider the point 2R (as in Figure 6B). Determine the maximum elevation encountered between the observer and position 2R. This is the horizon value at (X,Y), i.e., Z, where Z is the linear interpolation between the horizon at 1L and the horizon at 1R (See Appendix B1 for a discussion of the calculation of X,Y, & Z). Now project this Z value to the position 2R, obtaining Z'(2R) (as described in Appendix B2). In order to be seen, the target point must be higher than Z'(2R), so HOWHI(2R) = Z'(2R)-E(2R). In addition the vegetation must be considered. To guarantee the target is above the ground HOWHI(2R) must be the maximum of either HVEG(2R) or Z'(2R)-E(2R). Once HOWHI is determined the proper symbol is plotted for this point. Now the horizon at 2R must be computed. The horizon will simply be the maximum of (the projection of the old horizon to 2R) and (the current elevation, vegetation/urban included).

That is, HORIZON(2R) = MAX (Z'(2R), E(2R) + HVEG(2R))

Similarly, HOWHI(2L) = MAX (Z'(2L) - E(2L), HVEG(2L))

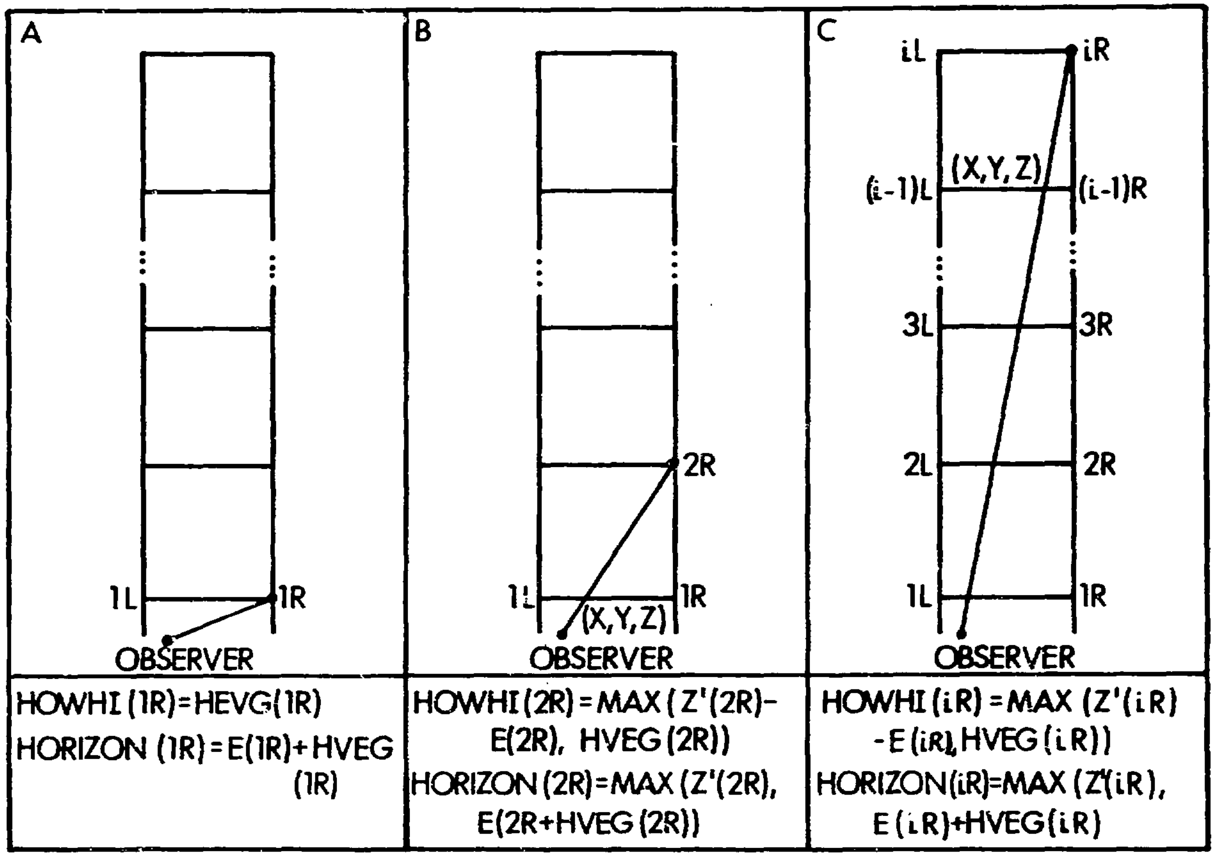
and HORIZON(2L) = MAX (Z'(2L), E(2L) + HVEG(2L).

In general, for the ith point above the observer (See Figure 6C)

HOWHI(i) = MAX (Z'(i) - E (i), HVEG(i))

and HORIZON(i) = MAX (Z'(i), E(i) + HVEG(i)).

Points south of the observer should be handled in a manner similar to those north of the observer, resulting in identical equations for HOWHI(i) and HORIZON(i).



• = TARGET POSITION
iL = ith POSITION ON LEFT SCAN
iR = ith POSITION ON RIGHT SCAN
E(iL) = DMA ELEVATION AT iL
Z = HORIZON AT (X,Y)
Z' (iL) = PROJECTION OF Z TO TARGET POSITION iL
HVEG (iL) = VEGETATION/URBAN HEIGHT AT iL

Figure 6. Calculation of HOWHI and HORIZON for Scans Closest to the Observer.

Scans East of Observer

Consider scanline R_2 in Figure 7A, consisting of points $(\ldots 0R_2, 1R_2, 2R_2, 3R_2, \ldots(n-1)R_2, nR_2)$. In order to obtain HOWHI's for this line, only the horizon values for scanline R and the terrain elevations for Scanline R_2 are needed. Working upward from the observer, the first target point to consider is $1R_2$. Check the terrain elevation at $1R_2$. This must be compared with the effect of the maximum interrupt encountered earlier (the horizon effect). The intersection of the observer-target line with the line of known horizon values closest to the target is at (X,Y). A linear interpolation between HORIZON(1R) and HORIZON(OR) will provide Z, the horizon height at (X,Y). Project Z to IR_2, obtaining Z' $(1R_2)$. Then

$$\text{HOWHI}(1R_2) = \text{MAX}\ (Z'(1R_2) - E(1R_2),\ \text{HVEG}(1R_2))$$

and $$\text{HORIZON}\ (1R_2) = \text{MAX}\ (Z'(1R_2),\ E(1R_2) + \text{HVEG}(1R_2)).$$

Similarly, using Figure 7B, the appropriate values for point $2R_2$ can be determined.

Figure 7C depicts a variation in this method. The observer-target line intersection with the line of known horizon values closest to the target is not along line R, but rather at a point (X,Y) between target points 2R and $2R_2$. In this case the HOWHI and HORIZON equations are as follows:

$$\text{HOWHI}(3R_2) = \text{MAX}(Z'(3R_2) - E(3R_2),\ \text{HVEG}(3R_2)).$$

$$\text{HORIZON}(3R_2) = \text{MAX}(Z'3R_2) + E(3R_2) + \text{HVEG}(3R_2)).$$

Thus the intersection of the observer-target line with the closest vertical horizon scan line is not necessarily the intersection of interest. The explanation in Appendix B1 for deriving X, Y and Z takes both of the cases into account.

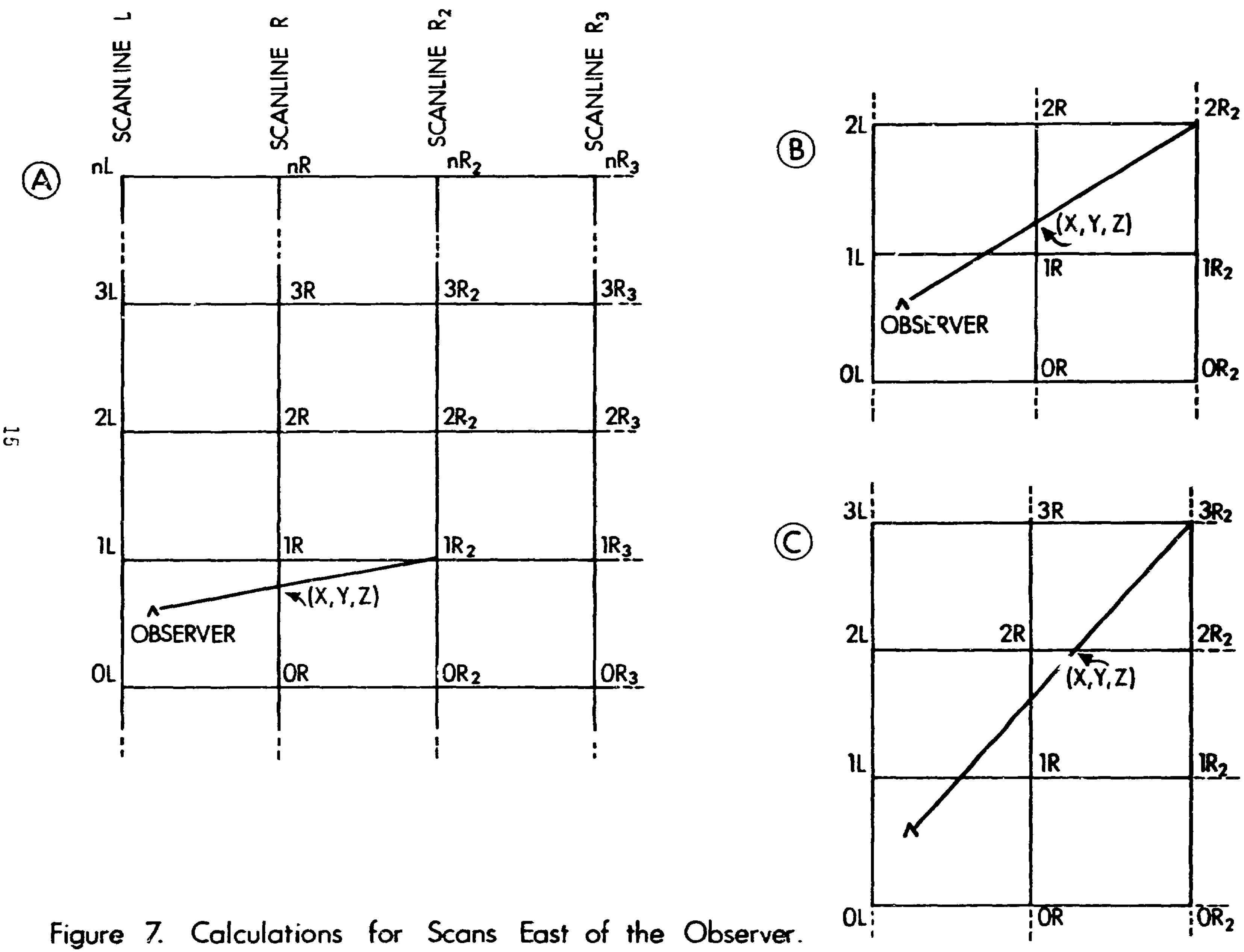


Figure 7. Calculations for Scans East of the Observer.

Scans West of Observer

In general, for points east of the observer

$$HOWHI(i) = MAX\ (Z'(i) - E(i),\ HVEG(i)\ )$$

$$HORIZON(i) = MAX\ (Z'(i),\ E(i) + HVEG(i)\ )$$

By simply reversing directions and starting with the horizon values corresponding to scanline L (calculated while working with scans closest to the observer) the same equations are found to apply to points west of the observer. Variations in the computation of values (X,Y,Z) in determining horizon effects are all accounted for in Appendix B1.

Algorithm Outine

In short, the map can be thought of as a lattice of equally spaced target positions. These are the points for which terrain elevations are available and can be thought of as south-to-north strings of target positions, or target scanlines. The following provides a general outline of the flow of the algorithm:

FIND AND PLOT LINE OF SIGHT INFORMATION FOR SCANS CLOSEST TO OBSERVER, WHILE CREATING ORIGINAL HORIZONS EAST AND WEST OF OBSERVER.

FOR SCANS FROM OBSERVER TO SIDE BOUNDARY (FIRST EASTWARD THEN WESTWARD DO:

READ ELEVATIONS FOR SCAN OF INTEREST

FOR POINTS FROM OBSERVER NORTH THEN FROM OBSERVER SOUTH DO:

> FIND INTERSECTION OF OBSERVER-TARGET LINE WITH HORIZON LINE CLOSEST TO TARGET
>
> INTERPOLATE BETWEEN KNOWN HORIZON VALUES TO OBTAIN HORIZON VALUE
>
> PROJECT HORIZON VALUE TO CURRENT TARGET POSITION (TAKING INTO ACCOUNT THE EFFECT OF EARTH CURVATURE)
>
> COMPARE CURRENT ELEVATION WITH PROJECTED HORIZON TO DETERMINE WHETHER LINE OF SIGHT EXISTS
>
> UPDATE HORIZON VALUE
>
> PLOT RESULTS

STOP

3.2 Solution to Storage and Compute Time Problem.

A. Storage. The savings in storage requirements for the SEE-FAR algorithm are considerable. The profile algorithm required all the elevation data to be easily accessible, since many scans of data were needed to produce each profile; the newer algorithm needs only the current scanline of elevation data and the most recently updated horizon information to perform the necessary computations. The amount of random access memory required, then, is a function of the north-to-south dimension of the map rather than the area of the map. A map that is 100 x 50 kilometers in size would take no more main memory than a map 1 x 50 kilometers, given equal grid size. The original algorithm would have taken 100 times more storage. Maps that would have been virtually impossible with the old algorithm are now quite possible. But if only storage requirements are reduced, the reader may wonder whether the machine time requirements would be prohibitive for production of these larger maps. The savings in computing time resulting from this approach must be considered.

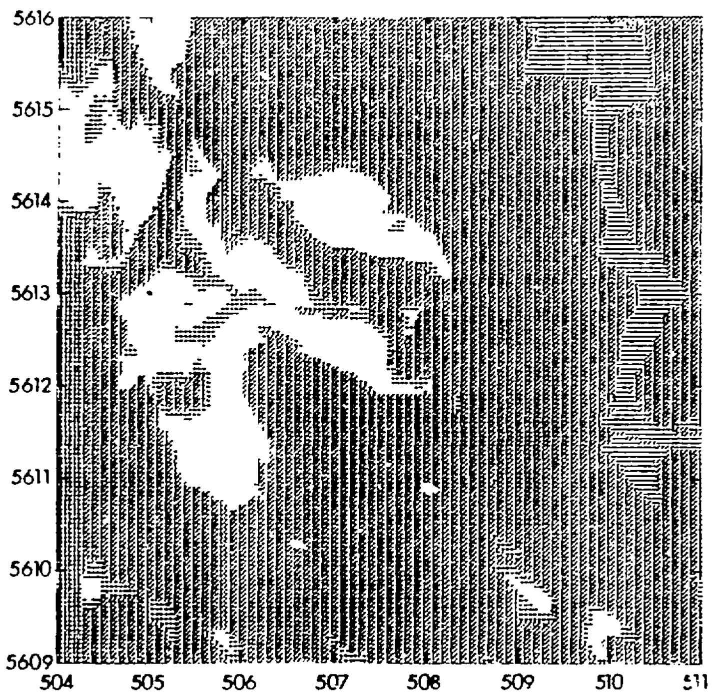
B. Computing Time. As mentioned earlier, the original algorithm required on the order of n^3 operations for an n x n sized map, because each profile consists of many points, and each of these points might have to be considered in a single line-of-sight calculation. The newer algorithm, however, simply compares the current target elevation with the horizon effect at the target point. Thus, a small fixed number of calculations are required for each target point on the map. This algorithm is, then, of the order n^2 for an n x n map. For maps of larger sizes the computing time savings will be considerable. It can in many instances be the difference between a map reasonably produced and one so expensive in computation cost that its value to the tactician or anlayst cannot outweigh the expense. Note that since n^2 target points are involved in an n x n map, order n^2 is the minimum possible complexity. A few examples of these differences are discussed in the following section.

4. COMPARISON OF MAPS RESULTING FROM EACH ALGORITHM

4.1 Discussion. The LOSMAP and SEEFAR algorithms discussed are quite different in nature. While each uses a method of linear interpolation to get results, the intuitive method interpolates between known elevations to get elevations along a profile, then checks to see if those elevations interrupt line of sight. The dynamic programming approach, on the other hand, keeps up with the effect of maximum interrupts encountered via the horizon array. A linear interpolation is made between horizon values to find a projection of the horizon to a specific target point. It is not surprising to discover the results of these methods are not always the same. But, while the "intuitive" LOSMAP method described uses interpolation and is therefore not an exact solution to the line of sight problem, it would be disappointing to find large discrepancies in the results of the two approaches. One would certainly hope the results of a new approach would conform somewhat to those of an intuitive method.

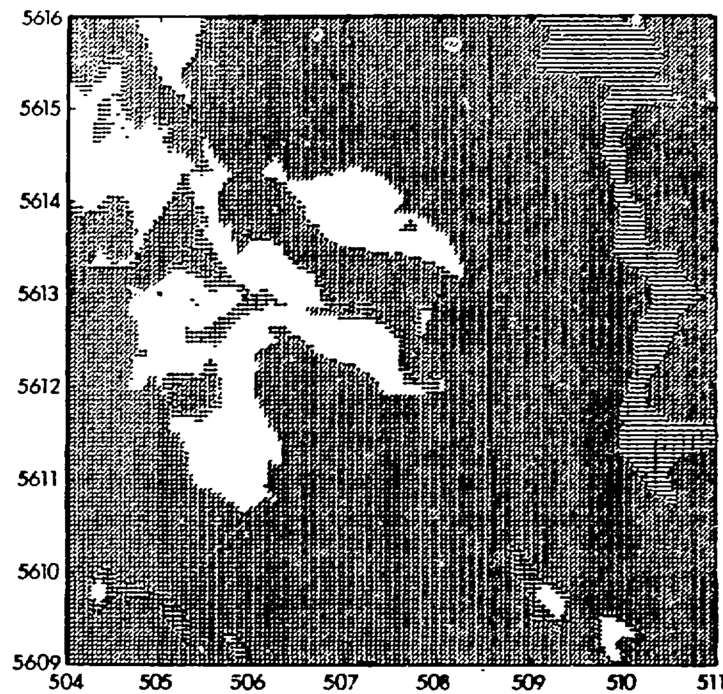
As it turns out, the results are quite similar. In fact, very close inspection is generally required to distinguish differences in line-of-sight maps with one target height. It is easier to detect differences when multiple target heights are used. The SEEFAR approach might at one time indicate a target visible that the LOSMAP approach indicated nonvisible. Another time the reverse situation might occur. In general, however, the boundaries of in-view and out-of-view areas do not differ widely between algorithms. It should be added that, while the dynamic programming approach may not be the first to come to mind when tackling the line-of-sight problem, it is a reasonable approach, one certainly not counter to intuition. And the savings in time and storage cannot be ignored.

4.2 Sample Maps. The following pages will provide the reader an opportunity to compare the results of the two algorithms.



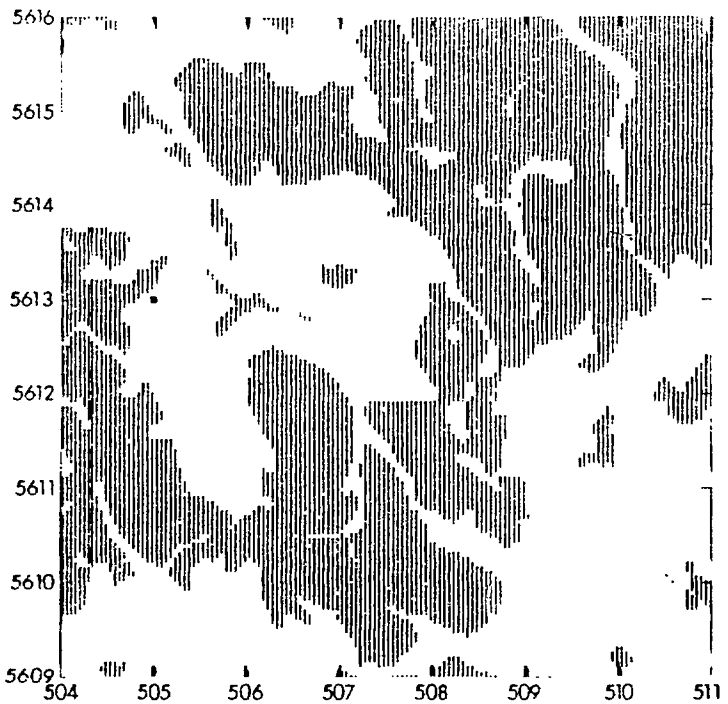
MAP OF POINTS VISIBLE TO A SINGLE OBSERVER
OBSERVER COORDINATES (505000.5613000)
OBSERVER HEIGHT = 25.00

Figure 8A. LOSMAP Algorithm.
Compute Time: 166 Sec.



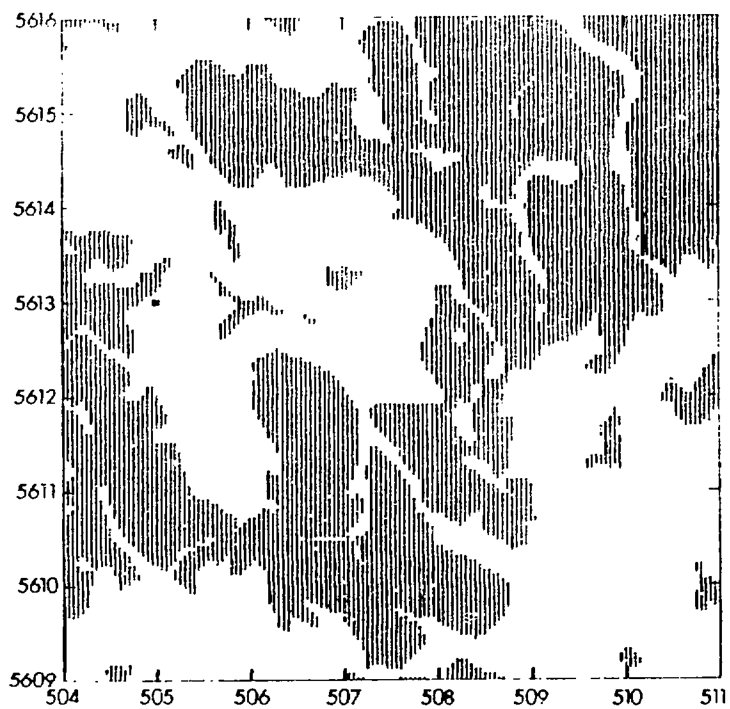
OBSERVER COORDINATES (505000. 5613000)
OBSERVER HEIGHT = 25.00
OBSERVER ID = 1

Figure 8B. SEEFAR Algorithm
Compute Time: 17 Sec.



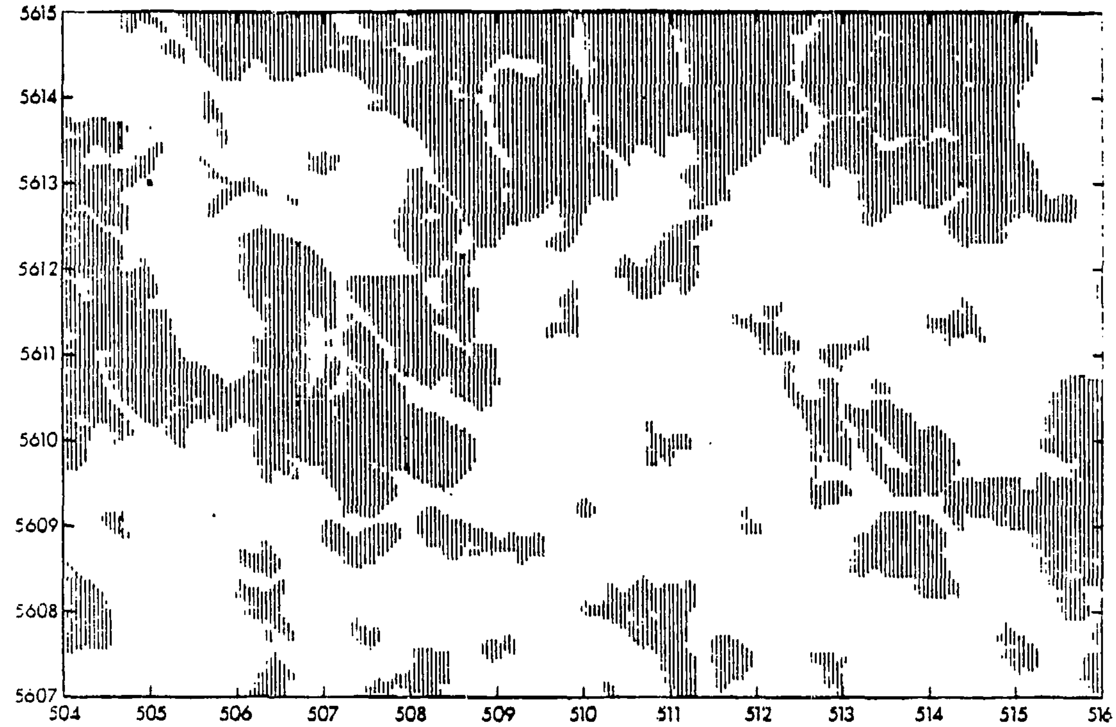
MAP OF POINTS VISIBLE TO A SINGLE OBSERVER
OBSERVER COORDINATES (505000, 5613000)
OBSERVER HEIGHT = 2000.00
TARGET HEIGHT = 2.00

Figure 9A. LOSMAP Algorithm.
Compute Time: 162 Sec.



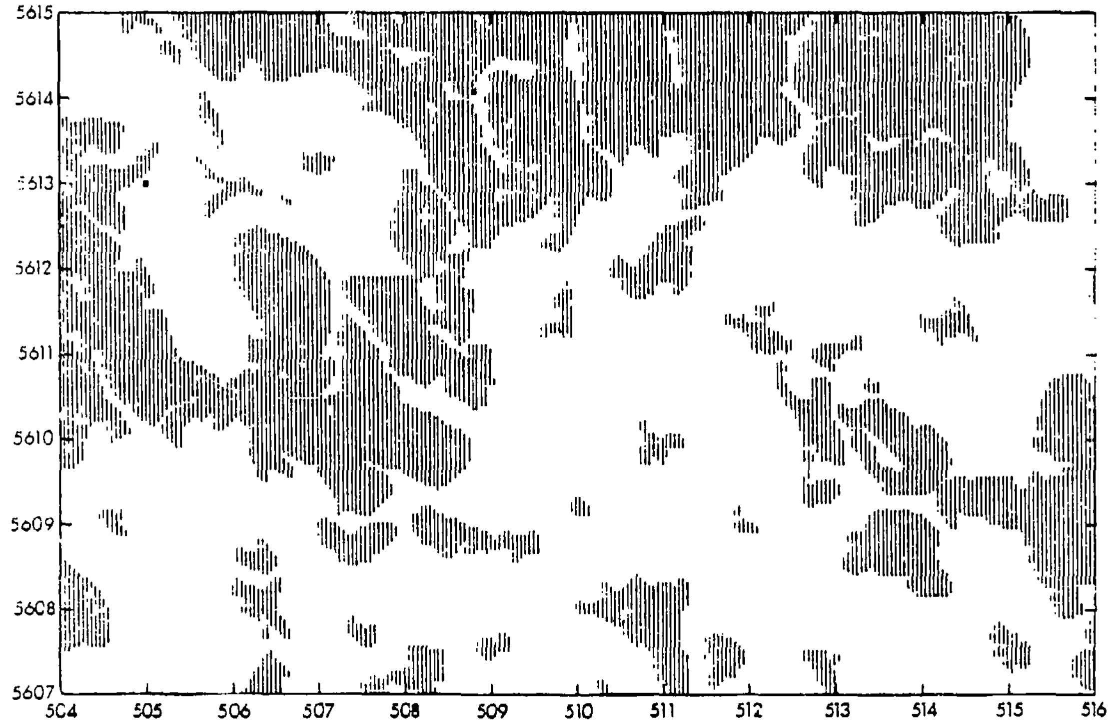
OBSERVER COORDINATES (505000, 5613000)
OBSERVER HEIGHT = 2000.00
OBSERVER ID = 1

Figure 9B. SEEFAR Algorithm.
Compute Time: 14 Sec.



MAP OF POINTS VISIBLE TO A SINGLE OBSERVER
OBSERVER COORDINATES (505000. 5613000)
OBSERVER HEIGHT = 2000.00
TARGET HEIGHT = 1.50

Figure 10A. LOSMAP Algorithm
Compute Time: 576 Sec.



OBSERVER COORDINATES (505000. 5613000)
OBSERVER HEIGHT = 2000.00
OBSERVER ID = 1

Figure 10B. SEEFAR Algorithm.
Compute Time: 20 Sec.

5. SUMMARY

5.1 Advantages of the SEEFAR Algorithm. The SEEFAR algorithm, then, provides a method for generating line-of-sight maps (and thereby obtaining intervisibility data) for a much larger area or for a smaller grid than could reasonably be provided using the LOSMAP algorithm. This dynamic programming approach results in an order of complexity of n^2 for an $n \times n$ map, which is the minimum that can be expected. Further, it reduces the storage requirements such that only one scanline of elevation data need be in memory at any one time. Thus, it is an order of magnitude better than the LOSMAP algorithm in both time and storage requirements.

5.2 The SEEFAR Program. Appendix A1 contains a copy of the SEEFAR program, an implementation of the algorithm discussed in this report. It should be noted that the algorithm discussion is concerned with the basic approach to the line-of-sight problem. The SEEFAR program has several embellishments. First, if an observer is in a forest/urban area, and the proper option is selected, SEEFAR will attempt to move the observer out of the obstruction. Second, the ratio of visible-to-total targets in a selected area of interest is given. Third, terrain data on as many as 12 DMA tapes can be merged and put into a direct access file. Finally, the HOWHI array, which indicates for every target position how high the target must be raised to be seen by the selected observer, can be saved on a permanent file. This allows one, through short fast-running programs, to vary target heights or merge line of sight data for several observers obtaining maps which indicate areas visible to m-out-of-n observers ($1 \leq m \leq n$). These m-out-of-n maps, or composite maps, are done by a separate program now, but could easily be provided via the SEEFAR program.

5.3 Problems Remaining. There are line-of-sight data requirements for which SEEFAR does not seem a viable alternative. For example, the LOSPATH program, in use at AMSAA, checks along paths for line of sight. To use SEEFAR for this problem one must generate a rectangular map containing all the path points and then check for line of sight at appropriate points within the rectangle. Because SEEFAR builds on information generated for all the points between the observer and the current target position, it does not seem optimal when target points are in scattered positions rather than at evenly spaced intervals within a rectangle of interest.

The question of how much discrepancy to expect between the LOSMAP and SEEFAR algorithms has not been determined analytically. This may be of interest; but since each approach provides an interpolated approximation to the line-of-sight problem, the value of such an analysis is questionable.

BIBLIOGRAPHY

Aylward, Graeme and Turnbull, Mark, "Visual Analysis: A Computer-Aided Approach to Determine Visibility", Computer-Aided Design, Vol. 9, 1977, pp 103-108.

Fleck, P.L., The Terrain Shielding Algorithm Used in the Mask Airborne Surveillance Radar Program, ESD-TR-79-276, Lincoln Laboratory, Lexington, MA.

Olson, Warren K., A Terrain Analysis of Four Tactical Situations, TM 158, U.S. Army Materiel Systems Analysis Activity, Aberdeen Proving Ground, MD.

Virbila, J.P., MASKPAS PROGRAMS Digitized Terrain, Flight Path Generator, Intervisibility for Evaluation of Air Defense Effectiveness, Vol 1, Joint Technical Coordinating Group for Munitions Effectiveness, U.S. Army Materiel Systems Analysis Activity, Aberdeen Proving Ground, MD.

The next page is blank.

APPENDIX A

SEEFAR USER'S AIDS

The next page is blank.

APPENDIX A1

SEEFAR PROGRAM

```
      PROGRAM SEEFAR(INPUT, OUTPUT, TAPE5=INPUT, TAPE6=OUTPUT,
     *                   TAPE7, TAPE8, TAPE9, TAPE10, TAPE11, TAPE12,
     *                    TAPE14, TAPE15, TAPE16, TAPE17, TAPE18, TAPE19,
     *                     TAPE2, TAPE3,
     *                      TAPE13)
C ***
C     BY BARBARA BROOME
C     PRODUCES LINE OF SIGHT MAP USING DYNAMIC PROGRAMMING ALGORITHM.
C     LOS DATA (THE HOWHI ARRAY) CAN BE STORE ON DISK FOR LATER WORK.
C     CALCULATES FRACTION OF (VISIBLE AREA) / (NON-VISIBLE AREA) FOR
C          RECTANGLE OF INTEREST.
C     TAPE13 HOLDS PLOT DATA
C     TAPE2  HOLDS MERGED TERRAIN DATA
C     TAPE3  HOLDS HOWHI DATA FOR FUTURE MAPS
C     TAPE5  HOLDS CARD INPUT
C     TAPE6  HOLDS PRINTED OUTPUT
C     ALL OTHERS ARE DMA TERRAIN DATA TAPES (PRE-MERGE)
C ***
      COMMON GRIDR
      COMMON /SEE/ XOBS, YOBS, HOBS, OBTRAN
      COMMON /AREA/ XORIGN, YORIGN, XSTOP, YSTOP
      COMMON /SCANS/ NX, NY
      COMMON /PLTSTF/ MAPID, SCALE, NPLTHT, PLTHT(10), IOBS, INK
      COMMON /BIG/ HOWHI(4000)
      COMMON /ASNSTF/ RNUM(10), DENOM
      COMMON /BOX/ BOXMIN, BOXMAX, BOYMIN, BOYMAX, IBOX
      COMMON /OTHERS/ IPNTNV, ITOTAV, XMIN, XMAX, YMIN, YMAX, NTAPES,
     *                    TGRID, ISAVE
      COMMON /VECIND/ JVEC
      REAL  LHOWHI(4000)
      COMMON /ZS/ ZOLD(4000), ZNOW(4000), ZLATER(4000)
      CALL RDMCHK
C ***
C     PUT DMA ELEVATIONS INTO DIRECT ACCESS FILE (UNIT 2).
C     MODIFY BOUNDARY COORDINATES TO CORRESPOND WITH DMA COORDINATES.
C     *EXPOSE OBSERVER AND FIND OBS. ELEVATION (HOBS NOT INCLUDED)
C ***
      CALL MERGIT(NTAPES, TGRID, GRIDR, XMIN, XMAX, YMIN, YMAX,
     *              NX, NY, XORIGN, YORIGN, XSTOP, YSTOP)
      CALL XPOSIT(XOBS,YOBS,IPNTNV)
      OBTRAN = ELVOBS(XOBS,YOBS,XORIGN,YORIGN,GRIDR)
      WRITE(6, 950) OBTRAN
C ***
C     PLOT PRELIMINARY INFO. (AXES, OBSERVER COORDINATES, ETC).
C     PLOT LOS INFO FOR SCANS CLOSEST TO OBSERVER
C     SAVE ZOLD IN LHOWHI - USE IT AS 1ST ZOLD FOR SCANS WEST OF OBS.
C     SAVE ZLATER         - USE IT AS 1ST ZOLD FOR SCANS EAST OF OBS.
C     IF SAVING MAP ON DISK, PUT PRELIM. INFO. IN REC 4001-4013 & SAVE
C ***
      XTOT = INT((XOBS-XORIGN+.00001)/GRIDR) * GRIDR + XORIGN + GRIDR
      CALL PLTPRE(XMIN, XMAX, YMIN, YMAX)
      IF (ISAVE .EQ. 1) CALL SAVPRE(ITOTAV)
      CALL FRSTIN(XTOT, LHOWHI, HOWHI, ICOL1)
      CALL PLTLN(LHOWHI, ICOL1-1, ITOTAV)
      CALL PLTLN(HOWHI, ICOL1, ITOTAV)
      IF (ISAVE .EQ. 1) CALL WRITMS(3,LHOWHI(1), NY, ICOL1-1)
```

```
      IF (ISAVE .EQ. 1) CALL WRITMS(3, HOWHI(1), NY, ICOL1)
      DO 200 I = 1, NY
         LHOWHI(I) = ZOLD(I)
  200    CONTINUE
C ***
C     FILL HOWHI & PLOT FOR EACH LINE EAST OF OBSERVER
C ***
      ICOL2 = ICOL1 + 1
      IF (ICOL2 .GT. NX) GO TO 450
      DO 400 ICOL = ICOL2, NX
         DO 300 I = 1, NY
            ZOLD(I) = ZLATER(I)
  300       CONTINUE
         CALL READMS(2, ZNOW(1), NY, ICOL)
         CALL ONELN(ICOL, HOWHI)
         CALL PLTLN(HOWHI, ICOL, ITGTAV)
         IF (ISAVE .EQ. 1) CALL WRITMS(3, HOWHI(1), NY, ICOL)
  400    CONTINUE
  450 CONTINUE
C ***
C     FILL HOWHI & PLOT FOR EACH LINE WEST OF OBSERVER
C ***
      ICOLM2 = ICOL1 - 2
      IF (ICOLM2 .LT. 1) GO TO 800
      ICOL = ICOL1 - 1
      DO 500 I = 1, NY
         ZLATER(I) = LHOWHI(I)
  500    CONTINUE
      DO 700 I = 1, ICOLM2
         ICOL = ICOL - 1
         DO 600 J = 1, NY
            ZOLD(J) = ZLATER(J)
  600       CONTINUE
         CALL READMS(2, ZNOW(1), NY, ICOL)
         CALL ONELN(ICOL, HOWHI)
         CALL PLTLN(HOWHI, ICOL, ITGTAV)
         IF (ISAVE .EQ. 1) CALL WRITMS(3, HOWHI(1), NY, ICOL)
  700    CONTINUE
  800 CONTINUE
      IF (IBOX .NE. 0) CALL PRFRSN
      CALL PLTPGE
      STOP
C ***
C     FORMATS
C ***
  950 FORMAT(' ELEVATION OF TERRAIN AT OBSERVER POSITION =', F10.1)
      END
```

```
      SUBROUTINE  DCDMAX ( X, Y, NREC)
C ***
C     BY MONTE COLEMAN
C     RETURNS X AND Y COORDINATES FROM DATA RECORD ALONG WITH RECORD NO.
C ***
      COMMON /DMACOM/ IBUF(700), ITEMP(80), ITEM
      DATA XSCALE/ 1.00 /, YSCALE / 1.00 /
C
      CALL UNPACK (IBUF, ITEMP, 10)
      NREC=ICVT(ITEMP(4), 3)
      X=FLOAT(ICVT(ITEMP(7),2))*XSCALE
      Y=FLOAT(ICVT(ITEMP(9),2))*YSCALE
      RETURN
      END
```

```
      SUBROUTINE DCTID (ISH, ISER, IED, XCOR, YCOR)
C ***
C     BY MONTE COLEMAN
C     DECODES A TAPE ID BLOCK
C     ISH    = MAP SHEET IDENTIFICATION
C     ISER   = MAP SERIES IDENTIFICATION
C     IED    = MAP EDITION IDENTIFICATION
C     CORNERS IN ORDER SW, NW, NE, SE
C ***
      COMMON /DMACOM/ IBUF(700), ITEMP(80), ITEM
      DIMENSION  XCOR(4), YCOR(4), ISH(2)
      CALL UNPACK (IBUF, ITEMP, 78)
      CALL RDTATR (ITEMP(7), 12)
      CALL PACK (ITEMP(7), ISH, 12)
      ISER=ICVT(ITEMP(67), 6)
      IED=ICVT(ITEMP(73), 6)
      IYC=19
      IXC=25
      DO 100 I= 1,4
         XCOR(I)=FLOAT(ICVT(ITEMP(IXC),6))*XSCALE
         YCOR(I)=FLOAT(ICVT(ITEMP(IYC),6))*YSCALE
         IXC=IXC+12
         IYC=IYC+12
  100    CONTINUE
      RETURN
      DATA  XSCALE / 1.00 /,  YSCALE / 1.00 /
      END
```

```
      FUNCTION ELVOBS(XOBS, YOBS, XORIGN, YORIGN, GRIDR)
C ***
C     BY BARBARA BROOME
C     FINDS THE OBSERVER ELEVATION
C ***
C
C          (X2,Y2,Z2)                      (X3,Y3,Z3)
C                        (XOBS,Y2,UP)
C               +.............+.............+
C               .                           .
C               .                           .
C               .                           .
C               .                           .
C               .      (XOBS,YOBS,ELVOBS)   .
C               .             +             .
C               .                           .
C               .                           .
C               .                           .
C               .                           .
C               .                           .
C               +.............+.............+
C                     (XOBS,Y1,DOWN)
C          (X1,Y1,Z1)                      (X4,Y4,Z4)
C
C ***
      DIMENSION RTSCN(4000)
      REAL LFTSCN(4000)
      COMMON /SCANS/ NX, NY
      ILEFT = INT((XOBS - XORIGN + .00001) / GRIDR) + 1
      IRIGHT = ILEFT + 1
      IBELOW = INT((YOBS - YORIGN + .00001) / GRIDR) + 1
      IABOVE = IBELOW + 1
C ***
      X1 = XORIGN + (ILEFT - 1) * GRIDR
      X2 = X1
      X3 = XORIGN + (IRIGHT - 1) * GRIDR
      X4 = X3
C ***
      Y1 = YORIGN + (IBELOW - 1) * GRIDR
      Y2 = YORIGN + (IABOVE - 1) * GRIDR
      Y3 = Y2
      Y4 = Y1
C ***
      CALL READMS(2, LFTSCN(1), NY, ILEFT)
      CALL READMS(2, RTSCN(1), NY, IRIGHT)
C ***
      Z1 = LFTSCN(IBELOW)
      Z1 = Z1 - TVEG(Z1)
      Z2 = LFTSCN(IABOVE)
      Z2 = Z2 - TVEG(Z2)
      Z3 = RTSCN(IABOVE)
      Z3 = Z3 - TVEG(Z3)
      Z4 = RTSCN(IBELOW)
      Z4 = Z4 - TVEG(Z4)
C ***
      DOWN = Z1 + ((XOBS - X1)/GRIDR) * (Z4-Z1)
```

```
      UP = Z2 + ((XOBS-X2)/GRIDR) * (Z3-Z2)
C ***
      ELVOBS = DOWN + ((YOBS-Y1)/GRIDR) * (UP-DOWN)
      RETURN
      END
```

```
      SUBROUTINE FDTATR (IA,N)
C ***
C     BY MONTE COLEMAN
C     CONVERTS FIELD DATA CHARACTERS TO DISPLAY CODE
C ***
      DIMENSION IA(1), IB(64)
      DO 100 I=1,N
         J=IA(I)+1
         IA(I)=ID(J)
  100    CONTINUE
      RETURN
C     FIELD DATA TO DISPLAY CODE TABLE
      DATA ID /   0, 49, 50,  0,  0, 45,  1,  2,
     *              3,  4,  5,  6,  7,  8,  9, 10,
     *              11, 12, 13, 14, 15, 16, 17, 18,
     *               19, 20, 21, 22, 23, 24, 25, 26,
     *                42, 38, 37, 58, 44, 59,  0, 43,
     *                 39, 41, 51,  0,  0,  0, 46,  0,
     *                  27, 28, 29, 30, 31, 32, 33, 34,
     *                   35, 36, 52,  0, 49, 47,  0,  0 /
      END
```

```
      SUBROUTINE FRCSEN(XTGT, YTGT, IFGSYM)
C ***
C     BY BARBARA BROOME
C     GETS NUMERATOR AND DENOMINATOR FOR CALCULATING FRACTION OF AREA
C     OF INTEREST SEEN BY OBSERVER - CONSIDERING VARIOUS TARGET HEIGHTS
C ***
      COMMON /PLTSTF/ MAPID, SCALE, NPLTHT, PLTHT(10), IOBS, INK
      COMMON /ASNSTF/ RNUM(10), DENOM
      COMMON /BOX/ BOXMIN, BOXMAX, BOYMIN, BOYMAX, IBOX
C ***
C     IF YOU AREN'T IN AREA OF INTEREST, DON'T DO ANYTHING - JUST RETURN
C ***
      IF ((XTGT .LT. BOXMIN) .OR. (XTGT .GT. BOXMAX) .OR.
     *    (YTGT .LT. BOYMIN) .OR. (YTGT .GT. BOYMAX)) RETURN
C ***
C     IF YOU CAN'T SEE ANY TARGETS, ADD 1 TO DENOMINATOR THEN RETURN
C ***
      IF (IFGSYM .GT. NPLTHT) DENOM = DENOM + 1
      IF (IFGSYM .GT. NPLTHT) RETURN
C ***
C     INCREASE NUMERATOR FOR EACH TARGET HEIGHT THAT CAN BE SEEN,
C     INCREASE DENOMINATOR, THEN RETURN
C ***
      DO 100 I = IFGSYM, NPLTHT
         RNUM(I) = RNUM(I) + 1
  100    CONTINUE
      DENOM = DENOM + 1
      RETURN
      END
```

```
      SUBROUTINE FFSTIM(XTGT, LHOWHI, HOWHI, ICOL1)
C ***
C     BY BARBARA BROOME
C     CALCULATES HOWHI FOR SCANS CLOSEST TO OBSERVER
C ***
      COMMON /ZS/ ZOLD(4000), ZNOW(4000), ZLATER(4000)
      COMMON /AREA/ XORIGN, YORIGN, XSTOP, YSTOP
      COMMON /SEE/ XOBS, YOBS, HOBS, OBTRAN
      COMMON /SCANS/NX, NY
      COMMON  GRIDR
      DIMENSION  HOWHI(1)
      REAL  LHOWHI(1)
C ***
C     FIND SCANS CLOSEST TO OBSERVER, LEFT IS ZOLD, RIGHT IS ZNOW
C ***
      ICOL1 = INT((XOBS - XORIGN + .00001) / GRIDR) + 2
      CALL READMS(2, ZNOW(1), NY, ICOL1)
      CALL READMS(2, ZOLD(1), NY, ICOL1-1)
C ***
C     FIND HOWHI FOR PTS. ABOVE OBSERVER ON BOTH SCANS
C ***
C *** FIRST TWO
      ITGT1 = INT((YOBS - YORIGN + .00001) / GRIDR) + 2
      IF (ITGT1 .GT. NY) GO TO 125
      XTGT = XORIGN + (ICOL1 - 1) * GRIDR
      YTGT = YORIGN + (ITGT1 - 1) * GRIDR
      HOWHI(ITGT1) = INT(TVEG(ZNOW(ITGT1)))
      ZLATER(ITGT1) = INT(ZNOW(ITGT1))
      LHOWHI(ITGT1) = INT(TVEG(ZOLD(ITGT1)))
      ZOLD(ITGT1) = INT(ZOLD(ITGT1))
      ITGTP1 = ITGT1 + 1
      IF (ITGTP1 .GT. NY) GO TO 125
      XTGT = XTGT - GRIDR
C *** ALL OTHERS
      DO 100 ITGT = ITGTP1, NY
          YTGT = YTGT + GRIDR
C *** RIGHT OF OBSERVER
          XTGT = XTGT + GRIDR
          ZP = ZPRIME(XTGT, YTGT, ITGT, ZOLD, ZLATER)
          HOWHI(ITGT) = ZP - ZNOW(ITGT) + TVEG(ZNOW(ITGT))
          ETVEG = INT(TVEG(ZNOW(ITGT)))
          HOWHI(ITGT) = AMAX1(HOWHI(ITGT), ETVEG)
          EZNOW = INT(ZNOW(ITGT))
          ZLATER(ITGT) = AMAX1(ZP, EZNOW)
C *** LEFT OF OBSERVER
          XTGT = XTGT - GRIDR
          ZP = ZPRIME(XTGT, YTGT, ITGT, ZLATER, ZOLD)
          LHOWHI(ITGT) = ZP - (ZOLD(ITGT) - TVEG(ZOLD(ITGT)))
          ETVEG = INT(TVEG(ZOLD(ITGT)))
          LHOWHI(ITGT) = AMAX1(LHOWHI(ITGT), ETVEG)
          LZOLD = INT(ZOLD(ITGT))
          ZOLD(ITGT) = AMAX1(ZP, LZOLD)
  100     CONTINUE
C ***
C     FIND HOWHI FOR PTS. BELOW OR LEVEL WITH OBS. ON BOTH SCANS
C ***
```

```
C *** FIRST TWO
  125 ITGTM1 = ITGT1 - 1
      XTGT = XORIGN + (ICOL1  - 1) * GRIDR
      YTGT = YORIGN + (ITGTM1 - 1) * GRIDR
C *** POINTS HORIZONTAL TO OBS., IF THEY EXIST
      IF (ABS(YTGT - YOBS) .GT. .00001) GO TO 150
      HOWHI(ITGTM1) = INT(TVEG(ZNOW(ITGTM1)))
      ZLATER(ITGTM1) = INT(ZNOW(ITGTM1))
      LHOWHI(ITGTM1) = INT(TVEG(ZOLD(ITGTM1)))
      ZOLD(ITGTM1) = INT(ZOLD(ITGTM1))
      ITGTM1 = ITGTM1 - 1
      XTGT = XORIGN + (ICOL1 - 1) * GRIDR
      YTGT = YORIGN + (ITGTM1 - 1) * GRIDR
      IF (ITGTM1 .LT. 1) GO TO 300
C *** FIRST TWO BELOW
  150 HOWHI(ITGTM1) = INT(TVEG(ZNOW(ITGTM1)))
      ZLATER(ITGTM1) = INT(ZNOW(ITGTM1))
      LHOWHI(ITGTM1) = INT(TVEG(ZOLD(ITGTM1)))
      ZOLD(ITGTM1) = INT(ZOLD(ITGTM1))
      ITGTM2 = ITGTM1 - 1
      ITGT = ITGTM1
      XTGT = XTGT - GRIDR
      IF (ITGTM2 .LT. 1) GO TO 300
C *** ALL OTHERS
      DO 200 J = 1, ITGTM2
         ITGT = ITGT - 1
         YTGT = YTGT - GRIDR
C *** RIGHT OF OBSERVER
         XTGT = XTGT + GRIDR
         ZP = ZPRIME(XTGT, YTGT, ITGT, ZOLD, ZLATER)
         HOWHI(ITGT) = ZP - (ZNOW(ITGT) - TVEG(ZNOW(ITGT)))
         ETVEG = INT(TVEG(ZNOW(ITGT)))
         HOWHI(ITGT) = AMAX1(HOWHI(ITGT), ETVEG)
         EZNOW = INT(ZNOW(ITGT))
         ZLATER(ITGT) = AMAX1(ZP, EZNOW)
C *** LEFT OF OBSERVER
         XTGT = XTGT - GRIDR
         ZP = ZPRIME(XTGT, YTGT, ITGT, ZLATER, ZOLD)
         LHOWHI(ITGT) = ZP - (ZOLD(ITGT) - TVEG(ZOLD(ITGT)))
         ETVEG = INT(TVEG(ZOLD(ITGT)))
         LHOWHI(ITGT) = AMAX1(LHOWHI(ITGT), ETVEG)
         EZOLD = INT(ZOLD(ITGT))
         ZOLD(ITGT) = AMAX1(ZP, EZOLD)
  200    CONTINUE
  300 RETURN
      END
```

```
      SUBROUTINE GOZIN2(RLITL,BIG,LITNAM,BIGNAM)
C ***
C     BY BARBARA BROOME
C     MAKES SURE RLITL GOES INTO BIG (WITH NO REMAINDER).
C     LITNAM = 'RLITL', BIGNAM = 'BIG'.
C ***
      IF ((AINT(BIG/RLITL)*RLITL) .NE. BIG) WRITE(6,100)
     *                          BIGNAM,LITNAM,BIGNAM,BIG,LITNAM,RLITL
      IF ((AINT(BIG/RLITL)*RLITL) .NE. BIG) STOP
      RETURN
  100 FORMAT(1X, A6, ' SHOULD BE DIVISIBLE BY ', A6, /,
     *      1X, A6, ' = ', F10.3, /,
     *       1X, A6, ' = ', F10.3)
      END
```

```
      FUNCTION  ICVT (IBUF, N)
C ***
C     BY MONTE COLEMAN
C     RETURNS INTERNAL INTEGER VALUE ASSOCIATED WITH N 6-BIT BYTES
C     STORED ONE PER WORD IN ARRAY IBUF.
C ***
      DIMENSION  IBUF(1)
      INTEGER SHIFT
      IF (N .GT. 10) GO TO 200
      ITEM=0
C
      DO 100 I=1,N
         ITEM=SHIFT(ITEM,6).OR.IBUF(I)
  100    CONTINUE
      ICVT=ITEM
      RETURN
C
  200 PRINT 210, N
      STOP
  210 FORMAT (' ICVT/ERROR.....LARGE N, N=', I10)
      END
```

```
      SUBROUTINE IMNMX(IVALUE,IMIN,IMAX,VALNAM,MINNAM,MAXNAM)
C ***
C     BY BARBARA BROOME
C     MAKES SURE IMIN IS LESS THAN IVALUE IS LESS THAN IMAX.
C     VALNAM = 'IVALUE', MINNAM = 'IMIN', MAXNAM = 'IMAX'.
C ***
      IF (IVALUE .LE. IMIN) WRITE(6,100) VALNAM,MINNAM,VALNAM,
     *                                     IVALUE,MINNAM,IMIN
      IF (IVALUE .GE. IMAX) WRITE(6,200) VALNAM,MAXNAM,VALNAM,
     *                                     IVALUE,MAXNAM,IMAX
      IF (IVALUE .LE. IMIN) STOP
      IF (IVALUE .GE. IMAX) STOP
      RETURN
  100 FORMAT(1X, A20,' SHOULD BE GREATER THAN ', A20,/,
     *        1X, A20,' = ', I10, /,
     *         1X, A20,' = ', I10)
  200 FORMAT(1X, A20,' SHOULD BE LESS THAN ',    A20,/,
     *        1X, A20,' = ', I10, /,
     *         1X, A20,' = ', I10)
      END
```

```
      FUNCTION LFRACT(Z,K)
C ***
C     BY ARTHUR GROVES
C         THIS FUNCTION RETURNS TEN TIMES THE FRACTIONAL PART OF THE
C         K-TH (OF THREE) ELEVATIONS PACKED INTO THE WORD Z.
C ***
      A=AINT(Z)
      C=A
      IF(K.EQ.1)GOTO 1
      A=10000.*(Z-A)
      C=AINT(A)
      IF(K.EQ.2)GOTO 1
      C=10000.*(A-C)
      C=AINT(C)
    1 C=.1*C + .001
      A=C-AINT(C)
      B=-.05
      DO 2 I=1,4
          B=B+.1
          IF(A.GT.B)GOTO 2
          LFRACT=I-1
          RETURN
    2     CONTINUE
      END
```

```
      FUNCTION LOCATE(X,XO,G)
C ***
C     BY ARTHUR GROVES
C     THIS FUNCTION RETURNS THE NUMBER OF GRID POINTS WHOSE ORDINATES
C     ARE LESS THAN OR EQUAL TO X.  XO (INPUT) IS THE ORDINATE OF THE
C     FIRST GRID POINT, AND G (INPUT) IS THE SPACING BETWEEN GRID POINTS
C ***
      LOCATE=AINT((X-XO)/G+1.)
      RETURN
      END
```

```
      SUBROUTINE MERGIT(NTAPES, TGRID, GRIDR, XMIN, XMAX, YMIN, YMAX,NX,
     $                  NY, XORIGN, YORIGN, XSTOP, YSTOP)
C ***
C     BY ARTHUR GROVES AND BARBARA BROOME
C     MERGE UP TO 12 DMA TAPES. ELEVATIONS GO IN DIRECT ACCESS ARRAY.
C ***
      DIMENSION MINDEX(4001)
      DIMENSION BUF(4000)
      DIMENSION L(12), XO(12), YO(12), R(8000), XC(4), YC(4)
      DIMENSION XL(12,5),YL(12,5),XCM(4),YCM(4),C(4)
      COMMON /VEGIND/ JVEG
      DATA (C(I),I=1,4)/'SOUTHWEST','NORTHWEST','NORTHEAST','SOUTHEAST'/
      CALL OPENMS(2, MINDEX, 4001, 0)
C ***
C     COMPUTE THINNING CONSTANT
C ***
      ISKIP = AINT(GRIDR/TGRID + .1)
      IBIG = 1000000000
C ***
C     READ HEADER RECORDS. COMPUTE INDIVIDUAL AND COMBINED TAPE START
C ***
      XSTART = 1000000000.
      YSTART = 1000000000.
      DO 100 I=1,NTAPES
         K = I + 6
         IF (I .GT. 6) K = I + 7
         CALL RDDATA(K, ICODE)
         CALL RTYPRC(ICODE, 1)
         CALL DCTID(ISH, ISER, IED, XC, YC)
            DO 50 N=1,4
               XL(I,N)=XC(N)
               YL(I,N)=YC(N)
   50          CONTINUE
            XL(I,5)=XC(1)
            YL(I,5)=YC(1)
         XO(I) = AMIN1(XC(1), XC(2))
         YO(I) = AMIN1(YC(1), YC(4))
         XSTART = AMIN1(XSTART, XO(I))
         YSTART = AMIN1(YSTART, YO(I))
         CALL RDDATA(K, ICODE)
         CALL RTYPRC(ICODE, 2)
         L(I) = 0
  100    CONTINUE
C ***
C     CHECK TO SEE IF EACH CORNER OF THE AREA TO BE MERGED
C     IS ON ONE OF THE TAPES.
C ***
      XCM(1)=XMIN
      XCM(2)=XMIN
      XCM(3)=XMAX
      XCM(4)=XMAX
      YCM(1)=YMIN
      YCM(2)=YMAX
      YCM(3)=YMAX
      YCM(4)=YMIN
      NCOT=0
      DO 175 K=1,4
```

```
            DO 150 I=1,NTAPES
                 NOR=0
                 DO 125 N=1,4
                      D=XCM(K)*(YL(I,N)-YL(I,N+1))
     $                 -YCM(K)*(XL(I,N)-XL(I,N+1))
     $                 -XL(I,N+1)*YL(I,N)+XL(I,N)*YL(I,N+1)
                      IF(D.LE.0.)NOR=NOR+1
 125                  CONTINUE
                 IF(NOR.EQ.4)GOTO 175
 150             CONTINUE
            NCOT=NCOT+1
            WRITE(6,40)C(K)
 175        CONTINUE
C ***
C      COMPUTE Y COORDINATE OF 1ST POINT ON EACH MERGED SCAN LINE
C ***
       Y = YSTART - TGRID
       DO 200 I=1,IBIG
            IF ((Y .LE. YMIN) .AND. (Y+TGRID .GT. YMIN)) GO TO 300
            Y = Y + TGRID
 200        CONTINUE
 300   YORIGN = Y
       ISAVE = I - 1
C ***
C      COMPUTE NUMBER OF POINTS ON EACH MERGED SCAN LINE
C ***
       DO 400 I=2,IBIG
            Y = Y + GRIDR
            IF ((Y-GRIDR .LE. YMAX) .AND. (Y .GT. YMAX)) GO TO 500
 400        CONTINUE
 500   NY = I
C ***
C      SET UP LOOP TO CYCLE THROUGH INCREASING VALUES OF X
C ***
       X = XSTART - TGRID
       XLAST = XSTART - GRIDR
       NX = 0
       DO 900 I=1,IBIG
            JIM = 0
            X = X + TGRID
            IF (X .GT. XMAX+GRIDR) GO TO 1000
C ***
C      CYCLE THRU THE TAPES PICKING OFF DATA FROM EACH TAPE
C      THAT CORRESPONDS TO THE CURRENT VALUE OF X
C ***
            DO 800 J=1,NTAPES
                 K = J + 6
                 IF (J .GT. 6) K = J + 7
                 IF ((X .LT. XO(J)-.1) .OR. (L(J).EQ.1)) GO TO 800
                 CALL RDDATA(K, ICODE)
                 IF ((ICODE.NE.'EOF') .AND. (ICODE.NE.'EOI')) GO TO 600
                 L(J) = 1
                 GO TO 800
 600             IF (ICODE .EQ. 'DATA') GO TO 700
                 IUNIT = J + 6
                 WRITE(6,10) IUNIT
                 STOP
```

```
 700          IF ((X.LT.XMIN-TGRID).OR.(X-XLAST.LT.GRIDR-1.))GO TO 800
              JIM = 1
              CALL DCDMAX(XX, YY, NREC)
              IY = YY
              M = AINT((YO(J)-YSTART)/TGRID + .1)
              CALL STDATA(R,M+IY+1,8000,JVEG)
 800          CONTINUE
C ***
C     THIN AND RECORD THE ELEVATION DATA FOR THIS SCAN LINE.
C     COMPUTE NUMBER OF SCANS AND X ORIGIN OF MERGED DATA.
C ***
          IF (JIM .EQ. 0) GO TO 900
          NX = NX + 1
          IF (NX .EQ. 1) XORIGN = X
          XLAST = X
          NSAVE = ISAVE + (NY-1)*ISKIP
          ICOUNT =1
          DO 850 J=ISAVE, NSAVE, ISKIP
               BUF(ICOUNT) = R(J)
              ICOUNT = ICOUNT + 1
 850          CONTINUE
          CALL WRITMS(2,  BUF(1), NY, NX)
 900      CONTINUE
C ***
C     WRITE IDENTIFYING DATA FOR MERGED TAPE
C ***
 1000 XSTOP = XORIGN + GRIDR*FLOAT(NX-1)
      YSTOP = YORIGN + GRIDR*FLOAT(NY-1)
      WRITE(6,20) NX, NY, XORIGN, YORIGN, XSTOP, YSTOP, GRIDR
      RETURN
C ***
C     FORMATS
C ***
   10 FORMAT(' SOMETHING NON-STANDARD ABOUT RECORD FORMAT ON UNIT', I10)
   20 FORMAT(//, ' MERGED ARRAY', /,
     $         ' NUMBER OF SCAN LINES =',                         I10,/,
     $         ' NUMBER OF POINTS PER SCAN LINE =',               I10,/,
     $         ' X COORDINATE OF FIRST SCAN LINE =',              F10.1,/,
     $         ' Y COORDINATE OF FIRST POINT ON EACH SCAN LINE =',F10.1,/,
     $         ' X COORDINATE OF LAST SCAN LINE =',               F10.1,/,
     $         ' Y COORDINATE OF LAST POINT ON EACH SCAN LINE =', F10.1,/,
     $         ' ARRAY GRID =',                                   F10.1)
   30 FORMAT(//, ' RECORD', I10, /,
     $       (' ', 10F7.1))
   40 FORMAT(/1X,A10,'CORNER OF AREA NOT ON ANY TAPE - RUN ABORTED
     $')
      END
```

```
      SUBROUTINE ONELN(ICOL, HOWHI)
C ***
C     BY BARBARA BROOME
C     GIVEN ARRAYS ZOLD (MAX PREVIOUS INTERRUPT) & ZNOW (CURRENT DMA
C     ELEVATIONS PLUS VEGETATION HEIGHT AND VEGETATION CODE)
C     CALCULATES FOR ONE SCAN HOW HIGH TGT MUST BE RAISED TO BE
C     SEEN & UPDATED MAX INTERRUPT (ZLATER).
C ***
      COMMON /ZS/ ZOLD(4000), ZNOW(4000), ZLATER(4000)
      COMMON /AREA/ XORIGN, YORIGN, XSTOP, YSTOP
      COMMON /SEE/ XOBS, YOBS, HOBS, OBTRAN
      COMMON /SCANS/ NX, NY
      COMMON GRIDR
      DIMENSION HOWHI(1)
C ***
C     SPECIAL CASE:  OBSERVER ON A SCAN LINE
C ***
      XTGT = XORIGN + (ICOL - 1) * GRIDR
      IF (ABS(XTGT - (XOBS-GRIDR)) .GT. .00001) GO TO 50
      ITGT1 = INT((YOBS - YORIGN + .00001) / GRIDR) + 2
      YTGT = YORIGN + (ITGT1 - 1) * GRIDR
      ITGTM1 = ITGT1 - 1
      ITGTM2 = ITGT1 - 2
      IF (ITGT1 .GT. NY) GO TO 25
      HOWHI(ITGT1) = INT(TVEG(ZNOW(ITGT1)))
      ZLATER(ITGT1) = INT(ZNOW(ITGT1))
   25 IF (ITGTM1 .LT. 1) GO TO 50
      HOWHI(ITGTM1) = INT(TVEG(ZNOW(ITGTM1)))
      ZLATER(ITGTM1) = INT(ZNOW(ITGTM1))
      IF (ITGTM2 .LT. 1) GO TO 50
      IF (ABS(YTGT - (YOBS+GRIDR)) .GT. .00001) GO TO 50
      HOWHI(ITGTM2) = INT(TVEG(ZNOW(ITGTM2)))
      ZLATER(ITGTM2) = INT(ZNOW(ITGTM2))
C ***
C     POINTS ABOVE OBSERVER
C ***
   50 ITGT1 = INT((YOBS - YORIGN + .00001) / GRIDR) + 2
      IF (ABS(XTGT - (XOBS-GRIDR)) .LT. .00001) ITGT1 = ITGT1 + 1
      XTGT = XORIGN + (ICOL - 1) * GRIDR
      YTGT = YORIGN + (ITGT1 - 1) * GRIDR
      IF (ITGT1 .GT. NY) GO TO 150
      DO 100 ITGT = ITGT1, NY
         ZP = ZPRIME(XTGT, YTGT, ITGT, ZOLD, ZLATER)
         HOWHI(ITGT) = ZP - (ZNOW(ITGT) - TVEG(ZNOW(ITGT)))
         ETVEG = INT(TVEG(ZNOW(ITGT)))
         HOWHI(ITGT) = AMAX1(HOWHI(ITGT), ETVEG)
         EZNOW = INT(ZNOW(ITGT))
         ZLATER(ITGT) = AMAX1(ZP, EZNOW)
         YTGT = YTGT + GRIDR
  100    CONTINUE
C ***
C     POINTS BELOW OBSERVER
C ***
  150 ITGTM1 = ITGT1 - 1
      IF (ABS(XTGT - (XOBS-GRIDR)) .LT. .00001) ITGTM1 = ITGT1 - 3
      YTGT = YORIGN + (ITGTM1 - 1) * GRIDR
```

```
      IF (ABS(YTGT - (YOBS-GRIDR)) .LT. .00001) ITGTM1 = ITGT1 - 4
      IF (ITGTM1 .LT. 1) GO TO 300
      ITGT = ITGTM1 + 1
      YTGT = YORIGN + (ITGT - 1) * GRIDR
      DO 200 I = 1, ITGTM1
         ITGT = ITGT - 1
         YTGT = YTGT - GRIDR
         ZP = ZPRIME(XTGT, YTGT, ITGT, ZOLD, ZLATER)
         HOWHI(ITGT) = ZP - (ZNOW(ITGT) - TVEG(ZNOW(ITGT)))
         ETVEG = INT(TVEG(ZNOW(ITGT)))
         HOWHI(ITGT) = AMAX1(HOWHI(ITGT), ETVEG)
         EZNOW = INT(ZNOW(ITGT))
         ZLATER(ITGT) = AMAX1(ZP, EZNOW)
200      CONTINUE
300   RETURN
      END
```

```
      SUBROUTINE PLTICS(IXRYAX, BOTRLF, START, STOP)
C ***
C     BY ARTHUR GROVES AND BARBARA BROOME
C     THIS ROUTINE PUTS LARGER TIC MARKS ALONG AN AXIS
C     IXRYAX - INDICATES WHETHER MARKING X- OR Y-AXIS (0-X , 1-Y)
C     BOTRLT - GIVES THE BOTTOM OR LEFT (DEPENDING ON IXRYAX) TIC
C              IF IXRYAX=0, GIVES BOTTOM; ELSE GIVES LEFT
C     START  - STARTING COORDINATE ON AXIS
C     STOP   - STOPPING COORDINATE ON AXIS
C ***
      DIMENSION EKS(2), WYE(2)
C ***
C     TO LABEL X-AXIS
C ***
      IF (IXRYAX .EQ. 1) GO TO 200
      WYE(1) = BOTRLF
      WYE(2) = BOTRLF + 60.
      DO 100 I = 1, 1000
         EKS(1) = START + 1000.*FLOAT(I-1)
         EKS(2) = EKS(1)
         IF ((EKS(1)-STOP) .GT. 50.) RETURN
         CALL PLTDTS(1, 0, EKS, WYE, 2, 0)
  100    CONTINUE
      RETURN
C ***
C     TO LABEL Y-AXIS
C ***
  200 EKS(1) = BOTRLF
      EKS(2) = BOTRLF + 60.
      DO 300 I = 1, 1000
         WYE(1) = START + 1000.*FLOAT(I-1)
         WYE(2) = WYE(1)
         IF ((WYE(1)-STOP) .GT. 50.) RETURN
         CALL PLTDTS(1, 0, EKS, WYE, 2, 0)
  300    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE PLTLN(HOWHI, ICOL, ITGTAV)
C ***
C     BY BARBARA BROOME
C     PLOTS ONE SCAN LINE OF HOWHI'S (ODD COLUMNS UP OTHERS DOWN)
C     IF KEEPING UP WITH FRAC. OF BOX SEEN, CALL NEC. SUBROUTINE
C ***
      DIMENSION XPLOT(2), YPLOT(2)
      DIMENSION SYMBOL(10)
      DIMENSION HOWHI(4000)
      COMMON /ZS/ ZOLD(4000), ZNOW(4000), ZLATER(4000)
      COMMON /AREA/ XORIGN, YORIGN, XSTOP, YSTOP
      COMMON /SCANS/ NX, NY
      COMMON GRIDR
      COMMON /PLTSTF/ MAPID, SCALE, NPLTHT, PLTHT(10), IOBS, INK
      COMMON /BOX/ BOXMIN, BOXMAX, BOYMIN, BOYMAX, IBOX
      DATA SYMBOL /'+>', '/>', '*>', 'A>', 'B>', 'C>', 'D>', 'E>', 'F>',
     *             '->'/
      SCLCNV = SCALE/100.
      HIGH = .45*GRIDR
      WIDE = .2808*GRIDR
      SIZE = .9*GRIDR / SCLCNV
      D = .5*GRIDR
      IF(MOD(ICOL, 2) .EQ. 0) D = -D
      XTGT = XORIGN + (ICOL-1)*GRIDR
      IF (NPLTHT .LE. 1) GO TO 400
C ***
C     CASE 1: MULTIPLE TARGET HEIGHTS
C     GIVEN SCAN OF HOWHI'S, FIGURE (FOR EACH Y) WHICH SYMBOL TO PLOT
C     AND PLOT IT
C ***
      DO 300 I = 1, NY
         IF (MOD(ICOL, 2) .EQ. 1) J = I
         IF (MOD(ICOL, 2) .EQ. 0) J = NY - I + 1
         YTGT = YORIGN + (J-1)*GRIDR
         DO 100 IFGSYM = 1, NPLTHT
            IF (ITGTAV .EQ. 1) HOWHI(J) = HOWHI(J) -
     *                     INT(TVEG(ZNOW(J)))
            IF (HOWHI(J) .LE. PLTHT(IFGSYM)) GO TO 200
 100        CONTINUE
         IFGSYM = 11
 200     IF(IFGSYM .GT. 1) CALL PLTSYM(SIZE, SYMBOL(IFGSYM-1), 0.,
     *                                 XTGT-WIDE, YTGT-HIGH)
         IF (IBOX .NE. 0) CALL FRCSEN(XTGT, YTGT, IFGSYM)
 300     CONTINUE
      RETURN
C ***
C     CASE 2: YES/NO MAP (ONE TARGET HEIGHT)
C     XPLOT(1)&(2) AND YPLOT(1)&(2) HOLD START/STOP OF O-O-V SEGMENTS
C     EVEN COLUMNS PLOT DOWN, ODD COLUMNS PLOT UP
C ***
 400  XPLOT(1) = XORIGN + (ICOL-1)*GRIDR
      XPLOT(2) = XPLOT(1)
      YPLOT(1) = 0.
      YPLOT(2) = 0.
      DO 500 I=1,NY
         J = I
```

```
      IF (MOD(ICOL,2) .EQ. 0) J = NY - I + 1
      YTGT = YORIGN + (J-1) * GRIDR
      IF (ITGTAV .EQ. 1) HOWHI(J) = HOWHI(J) -
     *         INT(TVEG(ZNOW(J)))
      IF (HOWHI(J) .LE. PLTHT(1)) IFGSYM = 1
      IF (HOWHI(J) .GT. PLTHT(1)) IFGSYM = 99999
      IF (IBOX .NE. 0) CALL FRCSEN(XTGT, YTGT, IFGSYM)
      IF ((YPLOT(1) .LE. 0.) .AND. (HOWHI(J) .GT. PLTHT(1)))
     *         YPLOT(1) = YORIGN + (J-1)*GRIDR - D
      IF (YPLOT(1) .LE. 0.) GO TO 500
      IF (HOWHI(J) .LE. PLTHT(1))
     *         YPLOT(2) = YORIGN + (J-1)*GRIDR - D
      IF (I .EQ. NY)
     *         YPLOT(2) = YORIGN + (J-1)*GRIDR + D
      IF (YPLOT(2) .LE. 0.) GO TO 500
      CALL PLTDTS(1, 0, XPLOT, YPLOT, 2, 0)
      YPLOT(1) = 0.
      YPLOT(2) = 0.
500   CONTINUE
    RETURN
    END
```

```
      SUBROUTINE PLTPRE(XMIN, XMAX, YMIN, YMAX)
C ***
C     BY BARBARA BROOME
C     PLOTS PRELIMINARY INFORMATION - BEFORE LOS CALCULATIONS
C ***
      DIMENSION PHTS(2)
      DIMENSION POBS(5)
      DIMENSION PLB(6)
      DIMENSION MID(2)
      DIMENSION LABELP(3), LABELI(4), SYMBOL(10)
      DIMENSION LGND1(1), LGND2(3), LGND3(3), LGND4(2), LGND5(3)
      COMMON /PLTSYS/ MAPID, SCALE, NPLTHT, PLTHT(10), IOBS, INK
      COMMON /SEE/ XOBS, YOBS, HOBS, OBTRAN
      DATA LABELP /'BROOME    ', 'BLDG 392  ', 'X2417     '/
      DATA LABELI /'BROOME    ', 'BLDG 392  ', 'X2417     ',
     *             'INK PLEASE'/
      DATA LGND1 /'LEGEND   >'/
      DATA LGND2 /'POSITION W', 'HERE TARGE', 'T        >'/
      DATA LGND3 /'X METERS H', 'IGH OR MOR', 'E        >'/
      DATA LGND4 /'CAN BE SEE', 'N, WHERE >'/
      DATA LGND5 /'SYMBOL    ', '          ', 'X =      >'/
      DATA SYMBOL/' >', '+>', '/>', '*>', 'A>', 'B>', 'C>', 'D>', 'E>',
     *            'F>'/
C ***
C     PLOT AXES
C ***
      PXMIN = 1000. * AINT(.001 * (XMIN + .001))
      PXMAX = 1000. * AINT(.001 * (XMAX - .001)) + 1000.
      PYMIN = 1000. * AINT(.001 * (YMIN + .001))
      PYMAX = 1000. * AINT(.001 * (YMAX - .001)) + 1000.
      F     = SCALE/100.
      XPAGE = (PXMAX - PXMIN)/F + 32.
      IF (XPAGE .LT. 46.) XPAGE = 46.
      YPAGE = (PYMAX - PYMIN)/F +  8.
      IF (YPAGE .LT. 21.) YPAGE = 21.
      CALL RMNMX(XPAGE,45.99,3657.,'XPAGE','TITLE PLOT','120 FT MAX')
      CALL RMNMX(YPAGE,20.9,73., 'YPAGE', 'LABEL SIZE', '29 IN. MAX')
      CALL PLTPGE
      IF (INK .EQ. 0) CALL PLTBEG(XPAGE, YPAGE, .3937, 13, LABELP)
      IF (INK .EQ. 1) CALL PLTBEG(XPAGE, YPAGE, .3937, 13,LABELI)
      CALL PLTSCA(6., 6., PXMIN, PYMIN, F, F)
      CALL PLTAXS(1000., 1000., PXMIN, PXMAX, PYMIN, PYMAX, 4)
C ***
C     INCREASE PLOT TICMARK SIZE (TOP, RT, BOTTOM, L)
C ***
      CALL PLTICS(0, PYMAX    , PXMIN, PXMAX)
      CALL PLTICS(1, PXMAX    , PYMIN, PYMAX)
      CALL PLTICS(0, PYMIN-60., PXMIN, PXMAX)
      CALL PLTICS(1, PXMIN-60., PYMIN, PYMAX)
C ***
C     LABEL AXES
C ***
      CALL LABELA(1000., 1000., PXMIN, PXMAX, PYMIN, PYMAX, .001, .001)
C ***
C     PLOT MAP IDENTIFICATION
C ***
```

```
      ENCODE(20, 900, MID(1)) MAPID
      CALL PLTSYM(.3, MID, 0., PXMIN, PYMAX + .8*F)
C ***
C     PLOT OBSERVER
C ***
      CALL PLTDTS(3, 0, XOBS, YOBS, 1, 0)
C ***
C     PLOT OBSERVER COORDINATES AND HEIGHT
C ***
      ENCODE(42, 910, POBS(1)) XOBS, YOBS
      CALL PLTSYM(.3, POBS, 0., PXMIN, PYMIN - 2.8*F)
      ENCODE(29, 920, POBS(1)) HOBS
      CALL PLTSYM(.3, POBS, 0., PXMIN, PYMIN - 3.6*F)
      ENCODE(29, 940, POBS(1)) IOBS
      CALL PLTSYM(.3, POBS, 0., PXMIN, PYMIN-4.4*F)
C ***
C     PLOT LEGEND
C ***
      CALL PLTSYM(.3, LGND1, 0., PXMAX+7.*F, PYMIN+12.*F)
      CALL PLTSYM(.3, LGND2, 0., PXMAX+7.*F, PYMIN+11.2*F)
      CALL PLTSYM(.3, LGND3, 0., PXMAX+7.*F, PYMIN+10.4*F)
      CALL PLTSYM(.3, LGND4, 0., PXMAX+7.*F, PYMIN+ 9.6*F)
      CALL PLTSYM(.3, LGND5, 0., PXMAX+7.*F, PYMIN+ 8.8*F)
      DO 100 I = 1, NPLTHT
         CALL PLTSYM(.3, SYMBOL(I), 0., PXMAX+8.*F,
     *                    PYMIN+(8.8-I*.8)*F)
         ENCODE(11,930,PHTS(1)) PLTHT(I)
         CALL PLTSYM(.3, PHTS, 0., PXMAX+13.*F,
     *                    PYMIN+(8.8-I*.8)*F)
  100    CONTINUE
      ENCODE(53,980,PLB(1))PLTHT(NPLTHT)
      CALL PLTSYM(.3, PLB, 0., PXMAX+8.*F, PYMIN+(8.8-I*.8)*F)
      RETURN
C ***
C     FORMATS
C ***
  900 FORMAT('MAP ID - ', A10, '>')
  910 FORMAT('OBSERVER C', 'OORDINATES', ' (', F7.0, ' , ', F8.0, ')>')
  920 FORMAT('OBSERVER H', 'EIGHT = ', F10.2, '>')
  930 FORMAT(F10.2, '>')
  940 FORMAT('OBSERVER I', 'D     = ', I10, '>')
  980 FORMAT('-     NO TARGET <=',F10.2,' METERS HIGH CAN BE SEEN>')
      END
```

```
      SUBROUTINE PRFRSN
C ***
C     BY BARBARA BROOME
C     PRINTS THE AVERAGE AREA SEEN FOR EACH TARGET HEIGHT
C ***
      COMMON /ASNSTF/ RNUM(10), DENOM
      COMMON /PLTSTF/ MAPID,SCALE, NPLTHT, PLTHT(10), IOBS, INK
      COMMON /BOX/ BOXMIN, BOXMAX, BOYMIN, BOYMAX, IBOX
      WRITE(6, 900) BOXMIN, BOYMIN, BOXMAX, BOYMAX
      WRITE(6, 910)
      DO 100  I = 1,  NPLTHT
         CFRACT = RNUM(I) / DENOM
         WRITE(6, 920) CFRACT, PLTHT(I)
  100    CONTINUE
      RETURN
C ***
C     FORMATS
C ***
  900 FORMAT(///, ' AREA OF INTEREST DEFINED BY ', /,
     *       ('     (', F10.1, ' ,', F10.1, ')  '))
  910 FORMAT(/, ' FRACTION OF AREA          TARGET', /,
     *          ' VISIBLE TO OBSERVER       HEIGHT',)
  920 FORMAT(/, '     ', F10.5, 7X, F10.1)
      END
```

```
      SUBROUTINE RDDATA(IU,ICODE)
C ***
C     BY MONTE COLEMAN
C     READS ONE BLOCK FROM A DMA TAPE ON UNIT NUMBER IU AND RETURNS
C     ICODE, WHICH INDICATES THE TYPE OF RECORD READ.
C ***
      COMMON /DMACOM/ IBUF(700),ITEMP(80),ITEM
      INTEGER TID,FID,DTA,EOF,EOI
      DATA MSKA / 77777777777700000000B/,MSKB/ 77777700000000000000B/,
     *       TID / 00000000002000000000B/,FID / 00000000120000000000B/,
     *        DTA / 41000000000000000000B/,EOF / 01254626150000000000B/,
     *         EOI / 01255131150000000000B/
      BUFFER IN (IU,1)(IBUF(1),IBUF(700))
      IEOF=UNIT(IU)
      IF(IEOF.GT.-1)GOTO 200
  100 ITEM=IBUF(1).AND.MSKA
      ICODE='  '
      IF(ITEM.EQ.TID)ICODE='TAPE ID'
      IF(ITEM.EQ.FID)ICODE='FILE ID'
      IF(ITEM.EQ.EOF)ICODE='EOF'
      IF(ITEM.EQ.EOI)ICODE='EOI'
      ITEM=IBUF(1).AND.MSKB
      IF(ITEM.EQ.DTA)ICODE='DATA'
      IF(ICODE.EQ.'  ')GOTO 240
      RETURN
  200 IF(IEOF.GT.0)GOTO 220
      PRINT 210,IU
      ICODE='EOF'
      RETURN
  210 FORMAT(' RDDATA/WARNING.....EOF ON UNIT',I5)
  220 PRINT 230,IU
      GOTO 100
  230 FORMAT(' RDDATA/WARNING.....PARITY ERROR ON UNIT',I5)
  240 PRINT 250,ITEM
      PRINT 260,IU
  260 FORMAT(' ERROR ON TAPE', I10)
      STOP
  250 FORMAT(' RDDATA/ERROR.....ILLEGAL INDICATOR'/10X,'ICODE= /',O20,
     *  '/')
      END
```

```
      SUBROUTINE RDNCHK
C ***
C     BY BARBARA BROOME
C     READ, WRITE, AND CHECK INPUT FOR SEEFAR PROGRAM
C           MAPID, SCALE, INK - TELLS WHAT MAP TITLE TO PUT ON PLOT,
C                               WHICH SCALE TO USE & WHETHER TO USE
C                               BALLPOINT PEN OR INDIA INK
C           IOBS, XOBS, YOBS, HOBS, IDNTMV - OBSERVER IDENTIFICATION,
C                               UTM COORDINATES, HEIGHT, AND INDICATION
C                                OF WHETHER TO ATTEMPT TO EXPOSE OBSERVER
C           NPLTHT, ITGTAV    - NO. TGT. HTS. CONSIDERED (LESS OR = 10)
C                               AND INDICATOR FOR WHETHER HTGT IS HT
C                               ABOVE VEG (IF 1) OR ABOVE LAND (IF 0)
C           PLTHT(I)          - I-TH TARGET HEIGHT (METERS)
C           XMIN, XMAX, YMIN, YMAX, GRIDR, JVEG
C                               UTM COORDINATES OF BOUNDARIES
C                               OF AREA TO BE CONSIDERED & GRID TO USE
C                               INDICATOR TO ADD VEGETATION
C                               VEG & URBAN=0,URBAN ONLY=1,NOTHING=2
C           NTAPES, TGRID     - NUMBER OF DMA TAPES REQUIRED & TAPE GRID
C           IBOX, ISAVE       - CALCULATE VISIBLE FRACTION OF BOX
C                               (1 = YES, 0 = NO)
C                               INDICATOR FOR WHETHER TO SAVE HOWHI'S ON
C                               DISK FOR FUTURE COMPOSITE CALCULATIONS
C                               (1 = YES, 0 = NO)
C                               HOWHI AFFECTED BY ITGTAV BEFORE STORING
C           BOXMIN, BOXMAX, BOYMIN, BOYMAX - IF IBOX = 1, DESCRIBES
C                               AREA OF INTEREST
C ***
      COMMON /VEGIND/ JVEG
      COMMON GRIDR
      COMMON /SEE/ XOBS, YOBS, HOBS, OBTRAN
      COMMON /PLTSTF/ MAPID, SCALE, NPLTHT, PLTHT(10), IOBS, INK
      COMMON /BOX/ BOXMIN, BOXMAX, BOYMIN, BOYMAX, IBOX
      COMMON /OTHERI/ IDNTMV, ITGTAV, XMIN, XMAX, YMIN, YMAX, NTAPES,
     *                TGRID, ISAVE
C ***
C     READ AND WRITE INPUT
C ***
      READ (5,900) MAPID, SCALE, INK
      WRITE(6,940) MAPID, SCALE, INK
      READ (5,910) IOBS, XOBS, YOBS, HOBS, IDNTMV
      WRITE(6,910) IOBS, XOBS, YOBS, HOBS, IDNTMV
      READ (5,960) NPLTHT, ITGTAV
      WRITE(6,960) NPLTHT, ITGTAV
      READ (5,920) (PLTHT(I), I = 1, NPLTHT)
      WRITE(6,920) (PLTHT(I), I = 1, NPLTHT)
      READ(5,930)XMIN,XMAX,YMIN,YMAX,GRIDR,JVEG
      WRITE(6,930)XMIN,XMAX,YMIN,YMAX,GRIDR,JVEG
      READ (5,910) NTAPES, TGRID
      WRITE(6,910) NTAPES, TGRID
      READ (5,960) IBOX, ISAVE
      WRITE(6,960) IBOX, ISAVE
      IF (IBOX .EQ. 1) READ (5,920) BOXMIN, BOXMAX, BOYMIN, BOYMAX
      IF (IBOX .EQ. 1) WRITE(6,920) BOXMIN, BOXMAX, BOYMIN, BOYMAX
C ***
```

```
C     DATA CHECKS
C ***
      CALL RMNMX(SCALE, 4999., 100001., 'SCALE', 'REAS MIN',
     *           'REAS MAX')
      CALL IMNMX(INK, -1, 2, 'INK', 'NEG NO',
     *           'REAS MAX')
      CALL RMNMX(XOBS, XMIN-.001, XMAX+.001, 'XOBS', 'XMIN', 'XMAX')
      CALL RMNMX(YOBS, YMIN-.001, YMAX+.001, 'YOBS', 'YMIN', 'YMAX')
      CALL RMNMX(HOBS, -.00001, 10001., 'HOBS', 'NEG. NO.',
     *           'REAS MAX')
      CALL IMNMX(IDNTMV, -1, 2, 'IDNTMV', 'NEG. NO.', 'REAS MAX')
      CALL IMNMX(NPLTHT, 0, 11, 'NO. OF PLOT HTS.', 'ZERO',
     *           'PLTHT DIM')
      CALL IMNMX(ITGTAV, -1, 2, 'ITGTAV', 'NEG. NO.', 'REAS MAX')
      DO 100 I = 1, NPLTHT
         CALL RMNMX(PLTHT(I), -.00001, 10001., 'NPLTHT(I)',
     *           'NEG NUMBER', 'REAS MAX')
  100    CONTINUE
      CALL RMNMX(XMIN, 99999., XMAX, 'XMIN', 'REAS MIN', 'XMAX')
      CALL RMNMX(XMAX, XMIN, 1000000., 'XMAX', 'XMIN', 'REAS MAX')
      CALL RMNMX(YMIN, 999999.,YMAX, 'YMIN', 'REAS MIN', 'YMAX')
      CALL RMNMX(YMAX, YMIN,10000000., 'YMAX', 'YMIN', 'REAS MAX')
      CALL RMNMX(GRIDR, 0., 1001., 'GRIDR', 'ZERO', 'REAS MAX')
      CALL IMNMX(JVEG,-1,3,'VEG ADD IN','NEG. NO','MAX VAL =2')
      CALL RMNMX(TGRID, 0., 1001., 'TGRID', 'ZERO', 'REAS MAX')
      CALL GOZIN2(TGRID, GRIDR, 'TGRID', 'GRIDR')
      CALL IMNMX(NTAPES, 0, 13, 'NTAPES', 'REAS MIN', 'MERGE LIM')
      CALL IMNMX(IBOX, -1, 2, 'IBOX', 'NEG. NO', 'REAS MAX')
      IF (IBOX .EQ. 1) CALL RMNMX(BOXMIN, XMIN-.001, BOXMAX, 'BOXMIN',
     *                            'XMIN', 'BOXMAX')
      IF (IBOX .EQ. 1) CALL RMNMX(BOXMAX, BOXMIN, XMAX+.001, 'BOXMAX',
     *                            'BOXMIN', 'XMAX')
      IF (IBOX .EQ. 1) CALL RMNMX(BOYMIN, YMIN-.001, BOYMAX, 'BOYMIN',
     *                            'YMIN', 'BOYMAX')
      IF (IBOX .EQ. 1) CALL RMNMX(BOYMAX, BOYMIN, YMAX+.001, 'BOYMAX',
     *                            'BOYMIN', 'YMAX')
      CALL IMNMX(ISAVE, -1, 2, 'ISAVE', 'NEG. NO', 'REAS MAX')
      RETURN
C ***
C     FORMATS
C ***
  900 FORMAT(A10, F10.0, I10)
  910 FORMAT(I10, 3F10.0, I10)
  920 FORMAT(8F10.0)
  930 FORMAT(5F10.0,I10)
  940 FORMAT(1X,A10,F10.0,I10)
  950 FORMAT(' ELEVATION OF TERRAIN AT OBSERVER POSITION =', F10.1)
  960 FORMAT(I10, I10)
      END
```

```
      SUBROUTINE RMNMX(ZVALUE,ZMIN,ZMAX,VALNAM,MINNAM,MAXNAM)
C ***
C     BY BARBARA BROOME
C     MAKES SURE ZMIN IS LESS THAN ZVALUE IS LESS THAN ZMAX.
C     VALNAM = 'ZVALUE', MINNAM = 'ZMIN', MAXNAM = 'ZMAX'.
C ***
      IF (ZVALUE .LE. ZMIN) WRITE(6,100) VALNAM,MINNAM,VALNAM,
     *                                    ZVALUE,MINNAM,ZMIN
      IF (ZVALUE .GE. ZMAX) WRITE(6,200) VALNAM,MAXNAM,VALNAM,
     *                                    ZVALUE,MAXNAM,ZMAX
      IF (ZVALUE .LE. ZMIN) STOP
      IF (ZVALUE .GE. ZMAX) STOP
      RETURN
  100 FORMAT(1X, A20,' SHOULD BE GREATER THAN ', A20,/,
     *        1X, A20,' = ', F10.0, /,
     *         1X, A20,' = ', F10.0)
  200 FORMAT(1X, A20,' SHOULD BE LESS THAN ',    A20,/,
     *        1X, A20,' = ', F10.0, /,
     *         1X, A20,' = ', F10.0)
      END
```

```
      SUBROUTINE RTYPRC (ICODE, I)
C ***
C *** BY MONTE COLEMAN
C *** CHECKS TO MAKE SURE INPUT RECORD IS THE RIGHT TYPE
C ***
      IF ((I .EQ. 1) .AND. (ICODE .NE. 'TAPE ID')) GOTO 100
      IF ((I .EQ. 2) .AND. (ICODE .NE. 'FILE ID')) GOTO 100
      IF (ICODE .EQ. 'EOF') GOTO 200
      IF (ICODE .EQ. 'EOI') GOTO 300
      IF ((I .GT. 2) .AND. (ICODE .NE. 'DATA')) GOTO 100
      IF (I .GT. 2500) GOTO 400
      RETURN
C ***
C *** THE FIRST 2 RECORDS SHOULD BE FILE ID AND TAPE ID
C ***
  100 WRITE(6,150) I,ICODE
  150 FORMAT('1RECORD ', I5, ' SHOULD NOT BE A ', A10, ' RECORD.')
      STOP
C ***
C *** IF EOF IS ENCOUNTERED
C ***
  200 WRITE(6,250) ICODE,I
  250 FORMAT(//,' ', A10, ' ENCOUNTERED AT RECORD ', I5)
      RETURN
C ***
C *** IF EOI IS ENCOUNTERED
C ***
  300 WRITE(6,250) ICODE,I
      STOP
C ***
C *** IF THERE SEEMS TO BE TOO MANY RECORDS ON THE TAPE
C ***
  400 WRITE(6,450)
  450 FORMAT('1ARE YOU SURE YOU SHOULD HAVE MORE THAN 2500 SCANLINES?')
      STOP
      END
```

```
      SUBROUTINE SAVPRE(ITOTAV)
C ***
C     BY BARBARA BROOME
C     SAVE PRELIMINARY FILE ID INFO IN RECORDS 4001-4013, IF ISAVE = 1.
C ***
      COMMON GRIDR
      COMMON /SCANS/ NX, NY
      COMMON /SEE/ XOBS, YOBS, HOBS, OBTRAN
      COMMON /AREA/ XORIGN, YORIGN, XSTOP, YSTOP
      COMMON /PLTSTH/ MAPID, SCALE, NPLTHT, PLTHT(10), IOBS, INK
      DIMENSION NINDEX(4015)
      CALL OPENMS(3, NINDEX, 4015, 0)
      CALL WRITMS(3, XOBS, 1, 4001)
      CALL WRITMS (3, YOBS, 1, 4002)
      CALL WRITMS (3, HOBS, 1, 4003)
      CALL WRITMS (3, XORIGN, 1, 4004)
      CALL WRITMS (3, YORIGN, 1, 4005)
      CALL WRITMS (3, XSTOP, 1, 4006)
      CALL WRITMS (3, YSTOP, 1, 4007)
      CALL WRITMS (3, NX, 1, 4008)
      CALL WRITMS (3, NY, 1, 4009)
      CALL WRITMS (3, GRIDR, 1, 4010)
      CALL WRITMS (3, MAPID, 1, 4011)
      CALL WRITMS (3, IOBS, 1, 4012)
      CALL WRITMS (3, ITOTAV, 1, 4013)
      RETURN
      END
```

```
      SUBROUTINE STDATA(Z,J,N,JVEG)
C ***
C      BY ARTHUR GROVES
C      STORE ELEVATIONS PLUS VEGETATION PLUS INDICATOR FRACTIONS IN
C      ARRAY Z STARTING AT Z(J).  N = DIMENSION OF Z-ARRAY.
C ***
      COMMON /DMACOM/ IBUF(700),ITEMP(60),ITEM
      DIMENSION Z(1),VEGCOD(4)
      INTEGER SHIFT
      DATA MSK1/10B/,MSK2/77777B/,VEGCOD/0.0,20.3,10.2,4.1/
      I=J
      IWORD=2
      IGET=1
      CALL UNPACK(IBUF(IWORD),ITEMP,60)
      IWORD=IWORD+6
  100 ISIGN=ITEMP(IGET).AND.MSK1
      IVEG=SHIFT(ITEMP(IGET),-4)
      ITEM=ICVT(ITEMP(IGET),3) .AND. MSK2
      IF(ITEM.EQ.MSK2)RETURN
      IF(ISIGN.NE.0)ITEM=-ITEM
      Z(I)=FLOAT(ITEM)
      IF(JVEG.EQ.0)Z(I)=Z(I)+VEGCOD(IVEG+1)
      IF(JVEG.EQ.1 .AND. IVEG.EQ.2)Z(I)=Z(I)+VEGCOD(3)
      I=I+1
      IF(I.GT.N)RETURN
      IGET=IGET+3
      IF(MOD(IGET,60).NE.1)GOTO 100
      CALL UNPACK(IBUF(IWORD),ITEMP,60)
      IWORD=IWORD+6
      IGET=1
      GOTO 100
      END
```

```
      FUNCTION TVEG(Z)
C ***
C     BY ARTHUR GROVES
C     RETURNS VEGETATION AMOUNT ADDED FOR THE GIVEN Z
C ***
      DATA ORCHAD /4.1/, URBAN /10.2/, FOREST /20.3/
      DPZ = Z - AINT(Z)
      TVEG = 0
      IF (DPZ .GT. .05) TVEG = ORCHAD
      IF (DPZ .GT. .15) TVEG = URBAN
      IF (DPZ .GT. .25) TVEG = FOREST
      RETURN
      END
```

```
      SUBROUTINE XPOSIT(X,Y,ICARE)
C ***
C     BY ARTHUR GROVES
C     THE PURPOSE OF THIS SUBROUTINE IS TO ATTEMPT TO MOVE AN OBSERVER
C     AT (X,Y) WHICH IS IN VEGETATION/URBAN TO A NEARBY POSITION WHICH
C     IS IN THE OPEN.  TO HAVE IT DO THIS, CALL THE SUBROUTINE WITH
C     ICARE=0.  IT THEN LOOKS AT THE ELEVATIONS AT THE FOUR CORNERS OF
C     THE SQUARE OF ELEVATIONS (X,Y) IS IN.  IF NONE OF THESE ARE
C     WOODED/URBAN, IT RETURNS THE ORIGINAL (X,Y).  IF ALL FOUR CORNERS
C     ARE WOODED/URBAN, OR IF EXACTLY ONE PAIR OF OPPOSITE CORNERS ARE
C     WOODED/URBAN, IT PRINTS AN APPROPRIATE STATEMENT AND STOPS THE
C     RUN, MAKING NO ATTEMPT TO FIND AN OPEN NEARBY LOCATION.  IF ONE,
C     TWO (ADJACENT) OR THREE CORNERS ARE WOODED/URBAN, IT MOVES THE
C     OBSERVER THE LEAST POSSIBLE DISTANCE TO AN ADJACENT SQUARE NOT
C     KNOWN TO HAVE ANY WOODED/URBAN CORNERS.  IF THIS NEW SQUARE IN
C     FACT HAS NO WOODED/URBAN CORNERS, IT RETURNS AS (X,Y) THE NEW
C     ADJUSTED OBSERVER LOCATION AND PRINTS THE NEW LOCATION.  IF THIS
C     NEW SQUARE DOES HAVE ONE OR MORE WOODED/URBAN CORNERS, IT PRINTS
C     THAT IT MADE AN UNSUCCESSFUL ATTEMPT TO MOVE THE OBSERVER AND
C     THEN STOPS THE RUN.  IF THE SUBROUTINE IS CALLED WITH ICARE=1,
C     THIS ENTIRE PROCEDURE IS BYPASSED AND (X,Y) IS RETURNED UNCHANGED
C     WHETHER OR NOT VEGETATION OR URBAN ARE PRESENT.
C ***
      DIMENSION T(4,4)
      COMMON GRIDR
      COMMON /SCANS/ NX,NY
      COMMON /AREA/ XORIGN, YORIGN,XSTOP,YSTOP
      COMMON /BIG/ SCAN(4000)
      LOGICAL L1,L2,L3,L4
    1 FORMAT(' ELEVATIONS REQUIRED FOR XPOZIT NOT PROVIDED BY MERGIT')
    2 FORMAT(' OBSERVER MOVED TO (',F10.1,',',F10.1,')')
    3 FORMAT(' OBSERVER ENTIRELY SURROUNDED BY VEGETATION OR URBAN AND C
     * OULD NOT BE MOVED')
    4 FORMAT(' EXACTLY ONE PAIR OF OPPOSITE CORNERS IN VEGETATION OR URB
     * AN - OBSERVER COULD NOT BE MOVED')
    5 FORMAT(' MADE ONE UNSUCCESSFUL ATTEMPT TO MOVE OBSERVER')
C ***
C     IF NO ATTEMPT TO MOVE THE OBSERVER IS TO BE MADE, RETURN.
C ***
      IF(ICARE.EQ.1)RETURN
C ***
C     IDENTIFY THE LOCATION OF THE 4X4 SQUARE OF ELEVATIONS SURROUNDING
C     THE OBSERVER.
C ***
      IL=LOCATE(X,XORIGN,GRIDR)
      JL=LOCATE(Y,YORIGN,GRIDR)
      XORIG = XORIGN + GRIDR * FLOAT(IL-2)
      YORIG = YORIGN + GRIDR * FLOAT(JL-2)
C ***
C     IF ALL 16 OF THESE ELEVATIONS HAVE NOT BEEN PROVIDED BY THE
C     MERGIT SUBROUTINE, PRINT THAT FACT AND STOP THE RUN.
C ***
      IF(IL-1.GE.1.AND.IL+2.LE.NX.AND.JL-1.GE.1.AND.JL+2.LE.NY)GOTO 100
      WRITE(6,1)
      STOP
C ***
```

```
C       FILL THE 4X4 ARRAY WITH ELEVATIONS.
C ***
  100 DO 120 I=1,4
         CALL READMS(2,SCAN(1),NY,IL+I-2)
         DO 110 J=1,4
            T(I,J)=SCAN(JL+J-2)
  110       CONTINUE
  120    CONTINUE
C ***
C       CLEAR THE 'SCAN' ARRAY, JUST IN CASE.
C ***
      DO 130 I=1,4000
         SCAN(I)=0.
  130    CONTINUE
C ***
C       SET 'MOVED' TO ZERO TO INDICATE THAT (X,Y) IS THE INPUT
C       OBSERVER POSITION AND NOT AN ADJUSTED POSITION.
C ***
      MOVED=0
C ***
C       DETERMINE WHICH CORNERS OF THE 2X2 SQUARE CONTAINING THE CURRENT
C       OBSERVER POSITION ARE IN VEGETATION OR URBAN.  THE APPROPRIATE
C       LOGICAL VARIABLE (L1, L2, L3 OR L4) WILL BE MADE 'TRUE' IF THE
C       CORRESPONDING CORNER IS IN VEGETATION OR URBAN.
C ***
  140 I=LOCATE(X,XORIG,GRIDR)
      J=LOCATE(Y,YORIG,GRIDR)
      XSCAN = XORIG + GRIDR * FLOAT(I-1)
      YSCAN = YORIG + GRIDR * FLOAT(J-1)
      L1 = AINT(10.*FRACT(T(I,J)+.00001)) .NE. 0.
      L2 = AINT(10.*FRACT(T(I+1,J)+.00001)) .NE. 0.
      L3= AINT(10.*FRACT(T(I+1,J+1)+.00001)) .NE. 0.
      L4 = AINT(10.*FRACT(T(I,J+1)+.00001)) .NE. 0.
C ***
C       IF NO CORNERS ARE IN VEGETATION OR URBAN, THE OBSERVER IS IN THE
C       OPEN. IF THIS IS THE ORIGINAL OBSERVER POSITION, RETURN TO THE
C       CALLING PROGRAM. IF THIS IS AN ADJUSTED POSITION, WRITE THAT FACT
C       AND THEN RETURN
C ***
      IF(L1.OR.L2.OR.L3.OR.L4)GOTO 150
      IF(MOVED.EQ.1)WRITE(6,2)X,Y
      RETURN
C ***
C       IF MOVED=1 INDICATING THAT THE OBSERVER HAS ALREADY BEEN MOVED
C       TO A NEW LOCATION AND IS STILL IN VEGETATION OR URBAN, WRITE
C       THIS FACT AND STOP THE RUN.
C ***
  150 IF(MOVED.EQ.0)GOTO 160
      WRITE(6,5)
      STOP
C ***
C       IF OBSERVER IS ENTIRELY SURROUNDED BY VEGETATION OR URBAN,
C       WRITE THIS FACT AND STOP THE RUN.
C ***
  160 IF(.NOT.(L1.AND.L2.AND.L3.AND.L4))GOTO 170
      WRITE(6,3)
      STOP
```

```
C ***
C      IF ONE PAIR OF OPPOSITE CORNERS ARE IN VEGETATION OR URBAN, WITH
C      THE OTHER PAIR IN THE OPEN, WRITE THIS FACT AND STOP THE RUN.
C ***
  170 IF(.NOT.((L1.AND.L3.AND..NOT.L2.AND..NOT.L4 .OR. L2.AND.L4.AND.
     *   .NOT.L1.AND..NOT.L3))GOTO 180
      WRITE(6,4)
      STOP
C ***
C      THE OBSERVER IS GOING TO BE MOVED TO A NEW LOCATION.  AT THIS
C      POINT MAKE  'MOVED' EQUAL 1  SO THAT WHEN CONTROL RETURNS TO
C      STATMENT 140 TO EXAMINE THE NEW LOCATION, IT WILL RECOGNIZE IT
C      AS A NEW LOCATION.
C ***
  180 MOVED=1
C ***
C      MOVE THE OBSERVER IF CORNER NUMBER 1 IS THE ONLY CORNER
C      IN VEGETATION OR URBAN.
C ***
      IF(.NOT.( L1.AND..NOT.L2.AND..NOT.L3.AND..NOT.L4))GOTO 190
      D1=XSCAN + GRIDR - X
      D2=YSCAN + GRIDR - Y
      IF(D1.LT.D2)X=XSCAN + GRIDR + .01
      IF(D1.GE.D2)Y=YSCAN + GRIDR + .01
      GOTO 140
C ***
C      MOVE THE OBSERVER IF CORNER NUMBER 2 IS THE ONLY CORNER
C      IN VEGETATION OR URBAN.
C ***
  190 IF(.NOT.(.NOT.L1.AND.L2.AND..NOT.L3.AND..NOT.L4))GOTO 200
      D1=X - XSCAN
      D2=YSCAN + GRIDR - Y
      IF(D1.LT.D2)X=XSCAN - .01
      IF(D1.GE.D2)Y=YSCAN + GRIDR + .01
      GOTO 140
C ***
C      MOVE THE OBSERVER IF CORNER NUMBER 3 IS THE ONLY CORNER
C      IN VEGETATION OR URBAN.
C ***
  200 IF(.NOT.(.NOT.L1.AND..NOT.L2.AND.L3.AND..NOT.L4))GOTO 210
      D1=X - XSCAN
      D2=Y - YSCAN
      IF(D1.LT.D2)X=XSCAN - .01
      IF(D1.GE.D2)Y=YSCAN - .01
      GOTO 140
C ***
C      MOVE THE OBSERVER IF CORNER NUMBER 4 IS THE ONLY CORNER
C      IN VEGETATION OR URBAN.
C ***
  210 IF(.NOT.(.NOT.L1.AND..NOT.L2.AND..NOT.L3.AND.L4))GOTO 220
      D1=XSCAN + GRIDR - X
      D2=Y - YSCAN
      IF(D1.LT.D2)X=XSCAN + GRIDR + .01
      IF(D1.GE.D2)Y=YSCAN - .01
      GOTO 140
C ***
C      MOVE THE OBSERVER IF CORNERS NUMBER 1 & 2 ARE THE ONLY
```

```
C     CORNERS IN VEGETATION OR URBAN.
C ***
  220 IF(.NOT.(L1.AND.L2.AND..NOT.L3.AND..NOT.L4))GOTO 230
      Y=YSCAN + GRIDR + .01
      GOTO 140
C ***
C     MOVE THE OBSERVER IF CORNERS NUMBER 1 & 4 ARE THE ONLY
C     CORNERS IN VEGETATION OR URBAN.
C ***
  230 IF(.NOT.(L1.AND..NOT.L2.AND..NOT.L3.AND.L4))GOTO 240
      X=XSCAN + GRIDR + .01
      GOTO 140
C ***
C     MOVE THE OBSERVER IF CORNERS NUMBER 2 & 3 ARE THE ONLY
C     CORNERS IN VEGETATION OR URBAN.
C ***
  240 IF(.NOT.(.NOT.L1.AND.L2.AND.L3.AND..NOT.L4))GOTO 250
      X=XSCAN - .01
      GOTO 140
C ***
C     MOVE THE OBSERVER IF CORNERS NUMBER 3 & 4 ARE THE ONLY
C     CORNERS IN VEGETATION OR URBAN.
C ***
  250 IF(.NOT.(.NOT.L1.AND..NOT.L2.AND.L3.AND.L4))GOTO 260
      Y=YSCAN - .01
      GOTO 140
C ***
C     MOVE THE OBSERVER IF CORNER 4 IS THE ONLY ONE IN THE OPEN.
C ***
  260 IF(.NOT.(L1.AND.L2.AND.L3.AND..NOT.L4))GOTO 270
      X=XSCAN - .01
      Y=YSCAN + GRIDR + .01
      GOTO 140
C ***
C     MOVE THE OBSERVER IF CORNER 3 IS THE ONLY ONE IN THE OPEN.
C ***
  270 IF(.NOT.(L1.AND.L2.AND..NOT.L3.AND.L4))GOTO 280
      X=XSCAN + GRIDR + .01
      Y=YSCAN + GRIDR + .01
      GOTO 140
C ***
C     MOVE THE OBSERVER IF CORNER 2 IS THE ONLY ONE IN THE OPEN.
C ***
  280 IF(.NOT.(L1.AND..NOT.L2.AND.L3.AND.L4))GOTO 290
      X=XSCAN + GRIDR + .01
      Y=YSCAN - .01
      GOTO 140
C ***
C     MOVE THE OBSERVER IF CORNER 1 IS THE ONLY ONE IN THE OPEN.
C     THIS IS THE ONLY CASE LEFT.
C ***
  290 X=XSCAN - .01
      Y=YSCAN - .01
      GOTO 140
      END
```

```
      FUNCTION ZPRIME(XTGT, YTGT, ITGT, ZOLD, ZLATER)
C ***
C     BY BARBARA BROOME
C     IF ZPRIME IS GREATER THAN THE TARGET ELEVATION, PREVIOUS TERRAIN
C     MUST BE BLOCKING THE VIEW.  TGT VISIBLE IFF HOWHI .LE. 0.
C     ZOLD   ARRAY CONTAINING ELEVATIONS FOR PREVIOUS SCAN (TRUMPED UP).
C     ZLATER WILL BE ZOLD FOR NEXT SCAN LINE
C     X,Y    COORDINATES OF INTERSECT OF OBS-TGT LINE WITH ZOLD SCAN.
C     Z      INTERPOLATED ELEVATION CORRESPONDING TO POINT (X,Y)
C     RE     RADIUS OF THE EARTH IN METERS
C     ZPRIME PROJECTION OF Z TO TGT POINT
C ***
      DIMENSION ZOLD(1), ZLATER(1)
      COMMON GRIDR
      COMMON /SEE/ XOBS, YOBS, HOBS, OBTRAN
      LOGICAL QUAD1U, QUAD1D, QUAD2U, QUAD2D, QUAD3U, QUAD3D,
     *        QUAD4U, QUAD4D
      RE = 6378329.
      DX = XTGT - XOBS
      DY = YTGT - YOBS
C ***
C     DEFINE X, Y, Z AFTER DETERMINING WHICH PORTION OF WHICH QUADRANT
C     YOUR TARGET IS IN (8 POSIBILITIES)
C ***
      QUAD1U = (DX .GE. 0.) .AND. (DY .GE. 0.) .AND. (DX .LT. DY)
      IF(.NOT. QUAD1U)GO TO 150
      X = DX*(DY-GRIDR)/DY + XOBS
      Y = YTGT - GRIDR
      Z = (1-DX/DY)*ZLATER(ITGT-1) + (DX/DY)*ZOLD(ITGT-1)
      GO TO 500
C ***
  150 QUAD1D = (DX .GE. 0.) .AND. (DY .GE. 0.) .AND. (DX .GE. DY)
      IF(.NOT. QUAD1D)GO TO 200
      X = XTGT - GRIDR
      Y = DY*(DX-GRIDR)/DX + YOBS
      Z = (1-DY/DX)*ZOLD(ITGT) + (DY/DX)*ZOLD(ITGT-1)
      GO TO 500
C ***
  200 QUAD2U = (DX .LT. 0.) .AND. (DY .GE. 0.) .AND. (-DX .LT. DY)
      IF(.NOT. QUAD2U)GO TO 250
      X = DX*(DY-GRIDR)/DY + XOBS
      Y = YTGT - GRIDR
      Z = (1+DX/DY)*ZLATER(ITGT-1) - (DX/DY)*ZOLD(ITGT-1)
      GO TO 500
C ***
  250 QUAD2D = (DX .LT. 0.) .AND. (DY .GE. 0.) .AND. (-DX .GE. DY)
      IF(.NOT. QUAD2D)GO TO 300
      X = XTGT + GRIDR
      Y = DY*(DX+GRIDR)/DX + YOBS
      Z = (1+DY/DX)*ZOLD(ITGT) - (DY/DX)*ZOLD(ITGT-1)
      GO TO 500
C ***
  300 QUAD3U = (DX .LT. 0.) . AND. (DY .LT. 0.) .AND. (DX .LT. DY)
      IF(.NOT. QUAD3U)GO TO 350
      X = XTGT + GRIDR
      Y = DY*(DX+GRIDR)/DX + YOBS
```

```
      Z = (1-DY/DX)*ZOLD(ITGT) + (DY/DX)*ZOLD(ITGT+1)
      GO TO 500
C ***
  350 QUAD3D = (DX .LT. 0.) .AND. (DY .LT. 0.) .AND. (DX .GE. DY)
      IF(.NOT. QUAD3D)GO TO 400
      X = DX*(DY+GRIDR)/DY + XOBS
      Y = YTGT + GRIDR
      Z = (1-DX/DY)*ZLATER(ITGT+1) + (DX/DY)*ZOLD(ITGT+1)
      GO TO 500
C ***
  400 QUAD4U = (DX .GE. 0.) .AND. (DY .LT. 0.) .AND. (DX .GT. -DY)
      IF(.NOT. QUAD4U)GO TO 450
      X = XTGT - GRIDR
      Y = DY*(DX-GRIDR)/DX + YOBS
      Z = (1+DY/DX)*ZOLD(ITGT) - (DY/DX)*ZOLD(ITGT+1)
      GO TO 500
C *** QUAD4D
  450 X = DX*(DY+GRIDR)/DY + XOBS
      Y = YTGT + GRIDR
      Z = (1+DX/DY)*ZLATER(ITGT+1) - (DX/DY)*ZOLD(ITGT+1)
C ***
  500 CONTINUE
C ***
C     CALCULATE THE PROJECTION OF Z ONTO THE TARGET POSITION
C ***
      RZ = SQRT((X-XOBS)**2 + (Y-YOBS)**2)
      RT = SQRT((XTGT-XOBS)**2 + (YTGT-YOBS)**2)
      B  = RE + HOBS + OBTRAN
      A = ((RE+Z)*COS(RZ/RE)-RE-HOBS-OBTRAN) / ((RE+Z)*SIN(RZ/RE))
      XPRIME = B / (COT(RT/RE)-A)
      YPRIME = COT(RT/RE)*B / (COT(RT/RE)-A)
      ZPRIME = SQRT(XPRIME**2 + YPRIME**2) - RE
      RETURN
      END
```

Appendix A2

INPUT FOR SEEFAR PROGRAM

CARD	INPUT	FORMAT	DESCRIPTION
1	MAPID, SCALE, INK	A10,F10.0, I10	MAPID: Alphanumeric map identification. Will be plotted at the top left corner of the map. SCALE: Indicates scale at which to plot map (e.g.,50000.will provide a map plotted at a 1:50000 scale). Subroutine RDNCHK requires 5000. $\leq$ SCALE $\leq$ 100000. Plot paper limits requires map to be less than 64CM high, less than 3657CM wide. INK: If INK = 0 Plot with ballpoint pen. If INK = 1 , plot with india ink.

Appendix A2(Cont'd.)

INPUT FOR SEEFAR PROGRAM

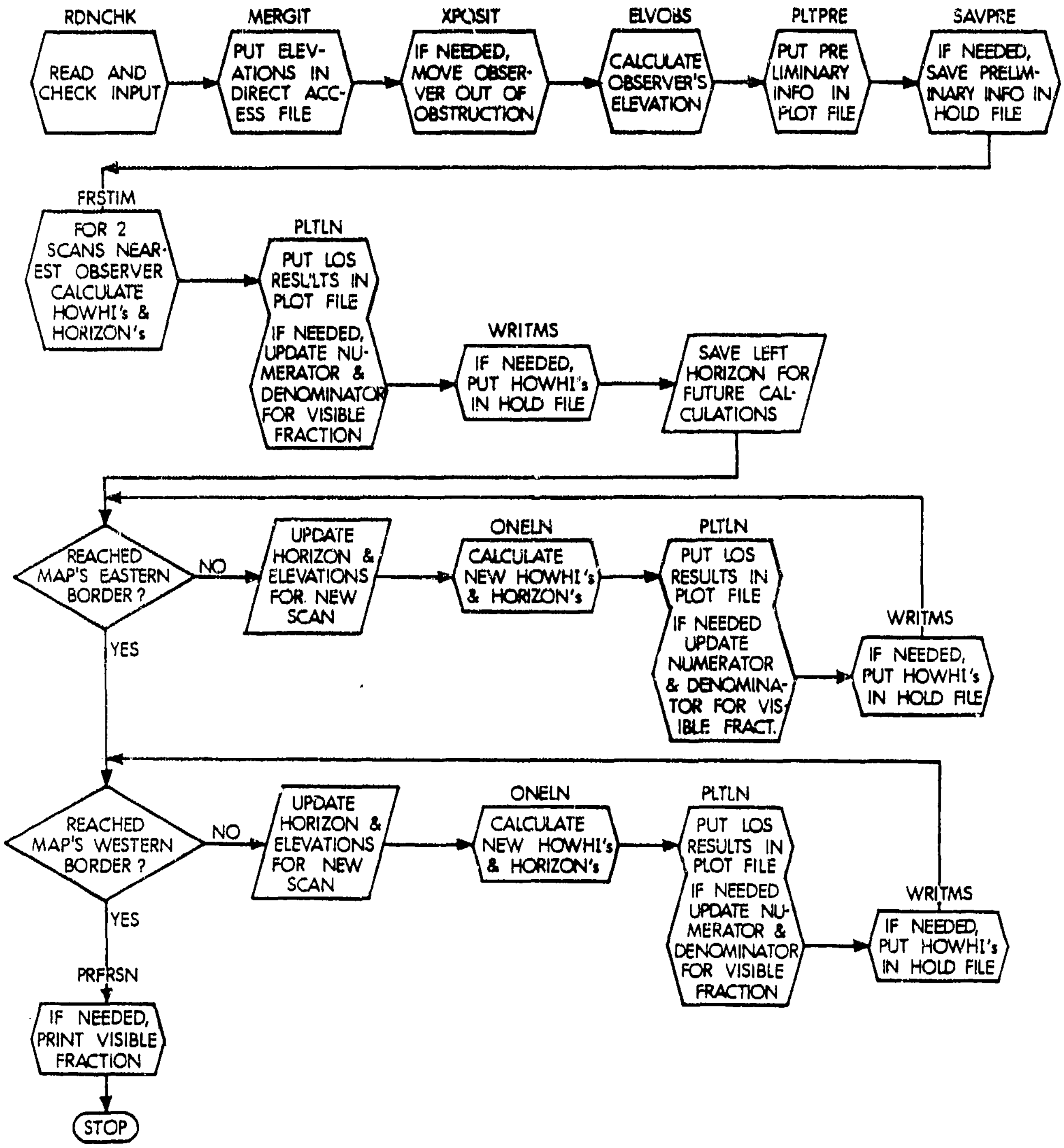
CARD	INPUT	FORMAT	DESCRIPTION
2	IOBS,XOBS,YOBS,HOBS,IDNTMV	I10,3F10.0,I10	IOBS: Observer identification number plotted below map. XOBS,YOBS: Observer UTM coordinates, Observer must be inside map boundaries. HOBS: Height of observer above terrain (in meters) subroutine RDNCHK requires $0 \leq HOBS \leq 10000$. IDNTMV: Indicates whether to attempt to move observer, should he be just inside a vegetation/urban boundary, to the outside, $0 \Rightarrow$ move if necessary, $1 \Rightarrow$ do not move.
3	NPLTHT,ITGTAV	2I10	NPLTHT: Number of target heights to be considered. Array PLTHT(I) size requires $1 \leq NPLTHT \leq 10$. ITGTAV: Indicates whether target height is the target's height above terrain only (ITGTAV=0) or above terrain and any vegetation/urban (ITGTAV=1).

CARD	INPUT	FORMAT	DESCRIPTION
4a-4b	PLTHT(I),I=1,NPLTHT	8F10.0	PLTHT(I): Lists the target heights (or plot heights) of interest, subroutine RDNCHK requires $0 \leq$ PLTHT (I) ≤ 10000.
5	XMIN,XMAX,YMIN,YMAX, GRIDR,JVEG	5F10.0, I10	XMIN,XMAX,YMIN,YMAX: The four corner UTM coordinates of the map to be plotted (all digits to the nearest meter included). GRIDR: The grid spacing of the resulting line of sight map. JVEG: Indicates whether to ignore vegetation/urban data or not. (0 $\Rightarrow$ include vegetation and urban heights, 1 $\Rightarrow$ include urban heights only, 2 $\Rightarrow$ include neither vegetation nor urban.) Note: STDATA routine is set to add 20 meters for forests (+.3 as indicator), 10 meters for urban (+.2 indicator) and 4 meters for orchards (+.1 indicator) via data statement

Appendix A2(Cont'd.)
INPUT FOR SEEFAR PROGRAM

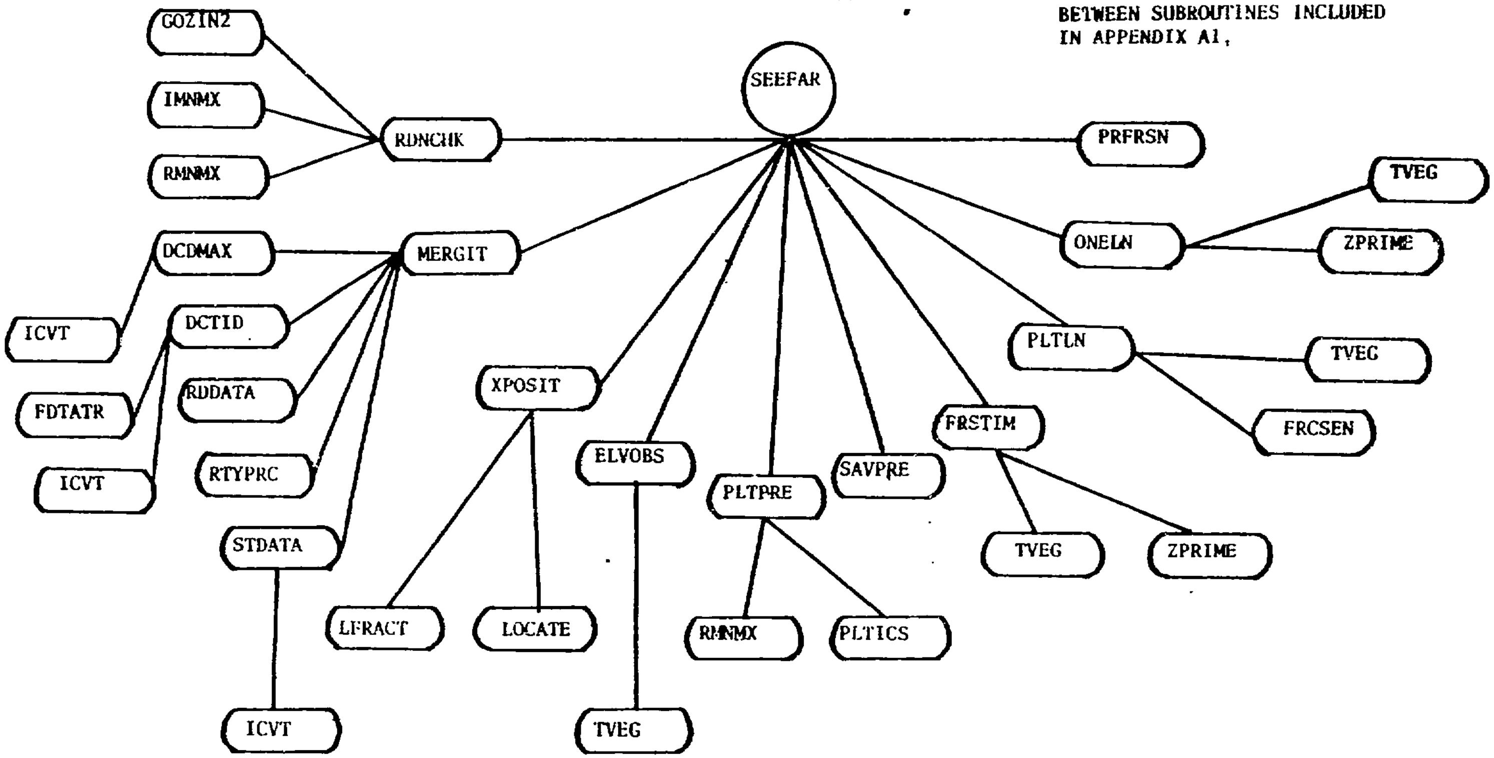
CARD	INPUT	FORMAT	DESCRIPTION
6	NTAPES, TGRID	I10,F10.0	NTAPES: The number of DMA tapes to be merged. Array dimensions in MERGIT require $1 \leq$ NTAPES ≤ 12 TGRID: The grid spacing of the input DMA terrain tapes, generally 12.5 or 25.
7	IBOX, ISAVE	2I10	IBOX: Indicates whether there is a box of interest (IBOX=1) for which one wishes to calculate the visible fraction or not (IBOX=0).
8 (Needed only if IBOX=1)	BOXMIN,BOXMAX,BOYMIN, BOYMAX	4F10.0	BOXMIN,BOXMAX,BOYMIN,BOYMAX: The four corner UTM coordinates for the area of interest referred to when IBOX = 1.

APPENDIX A3
SEEFAR FLOWCHART



APPENDIX A4

PROGRAM SKELETON: ILLUSTRATES THE RELATIONSHIP BETWEEN SUBROUTINES INCLUDED IN APPENDIX A1,



IN ADDITION TO THESE SUBROUTINES THE FOLLOWING SYSTEM SOFTWARE WAS USED:

PACK	-	packs several words into one
UNPACK	-	unpacks word into 1 character words
SHIFT	-	shifts bits in a word
UNIT	-	checks status (EOF, parity error, ready) of specified unit
READMS	-	reads direct access block
WRITMS	-	write direct access block
OPENMS	-	prepare for direct access reads/writes

PLTPGE	-	start a new plotting page
LABELA	-	label axes
PLTAXS	-	plot axes
PLTBEG	-	begin plot routines
PLTDTS	-	plot data specified
PLTSCA	-	set plotting scale
PLTSYM	-	plot symbol

Next page is blank.

APPENDIX B

SEEFAR ANALYST'S AIDS

The next page is blank.

Appendix B1

Calculation of Critical Horizon Coordinates

In determining whether a target position is in-view or out-of-view the first step is to consider the effect of intervening terrain. This is done by finding where the observer-target line intersects the closest line of known horizon values (as illustrated in Figure B1-1). This intersection location is referred to as (X,Y). To find the horizon value, Z, at (X,Y), one need only interpolate between the closest horizon values on the horizon scan, pictured as Z_1, Z_2.

The problem of solving for X,Y, and z breaks into eight cases (illustrated in Figure B1-2). These cases are determined by the relationship between the coordinates of the target and those of the observer.

Considering a pair of axes with origin at the observer position, each quadrant has been divided into two sectors. Targets in one sector have an observer-target line slope whose absolute value is less than or equal to one. In the other sector the absolute value of this slope is greater than one. The reason for this division will be explained by the following two examples.

Selecting quadrant 1, X, Y, and Z will be determined for the area where XTGT $\geq$ XOBS, YTGT $\geq$ YOBS and (XTGT - XOBS) $\geq$ (YTGT - YOBS). (See Figure B1-3.)

Letting GRID be the grid of the map data X = XTGT - GRID, since the horizon is one scan away. Knowing X, we can solve for Y from the equation of the observer-target line:

$$Y = \frac{(YTGT-YOBS)\ (XTGT-XOBS-GRID)}{(XTGT-XOBS)} + YOBS.$$

It remains to solve for Z. Consider the target to be the ITGTth position on a scan and ZOLD to be an array containing old horizon values.

Let DX = XTGT-XOBS and DY = YTGT-YOBS.

Then linearly interpolating,

$$Z = (1 - \frac{DY}{DX})\ ZOLD\ (ITGT) + (\frac{DY}{DX})\ ZOLD\ (ITGT-1)$$

Now consider a target in the upper portion of quadrant 1, where XTGT $\geq$ XOBS, TYGT $\geq$ YOBS and (XTGT-XOBS) < (YTGT - YOBS). See Figure B1-4.) In this example the closest line of known horizon values is formed by ZOLD (ITGT-1) and ZLATER (ITGT-1) rather than by two old horizon values. In this case

$$Y = YTGT - GRID$$

$$X = \frac{DX\ (DY-GRID) + XOBS}{DY}$$

$$Z = (1 - \frac{DX}{DY})\ \ ZLATER\ (ITGT-1) + (\frac{DX}{DY})\ ZOLD\ (ITGT-1)$$

Where DX, DY and GRID are as defined in the previous example. Similarly, equations for X, Y, and Z may be derived for each of the eight sectors, as indicated in Figure B1-5.

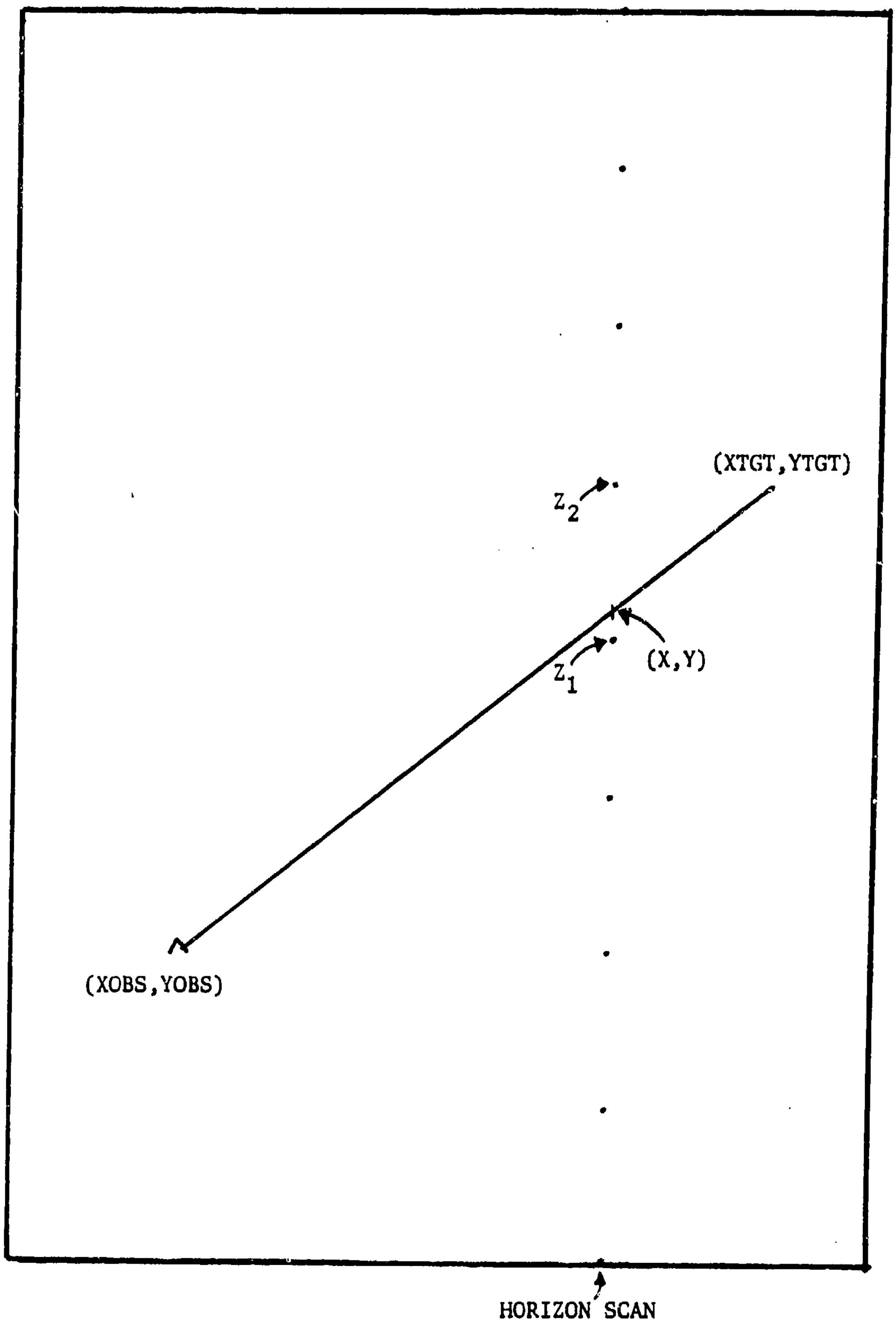


Figure B1-1
Locating Intersection of Observer - target line with
Closest Horizon Values

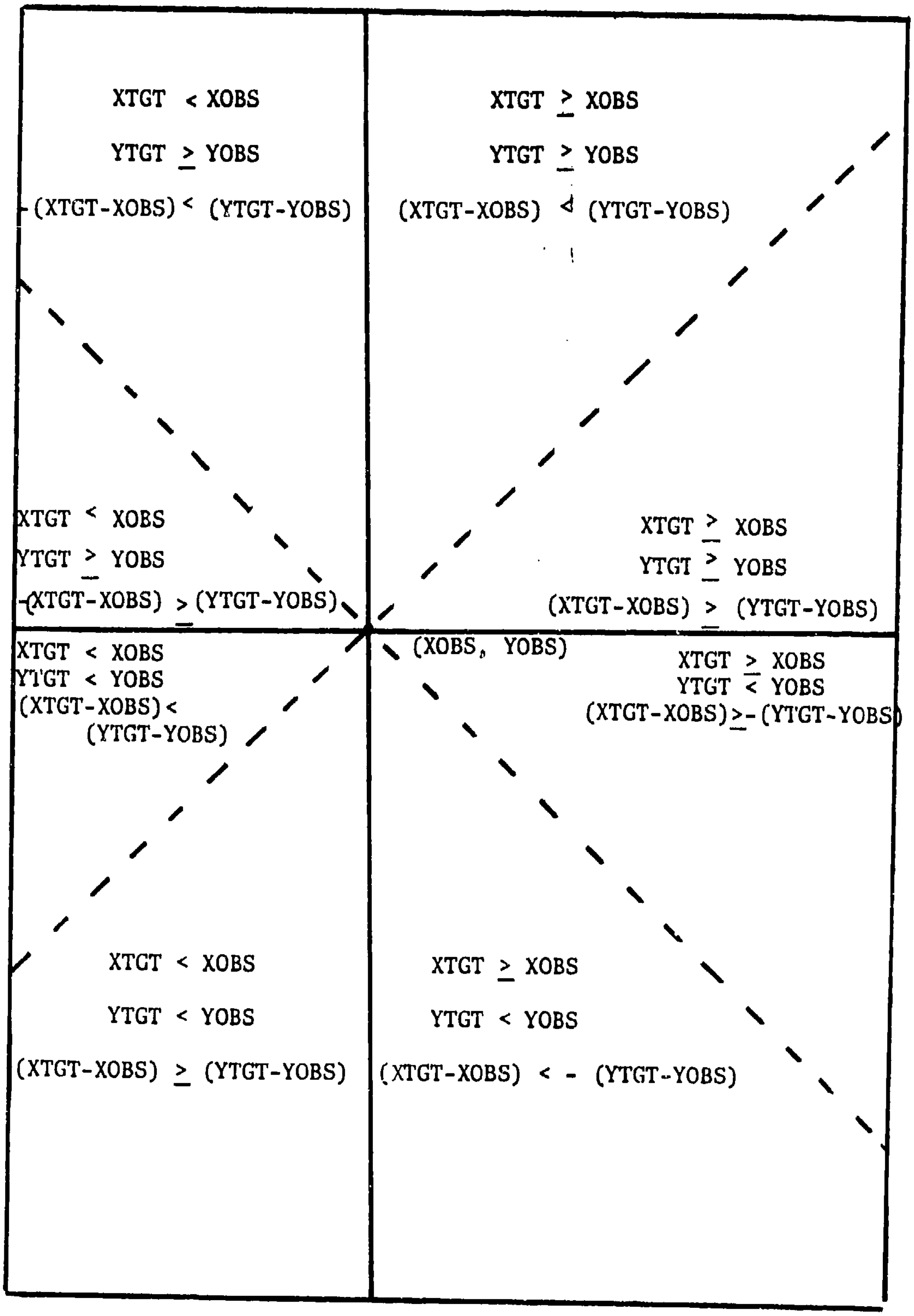


Figure B1-2
Eight Cases for Computing X,Y,Z

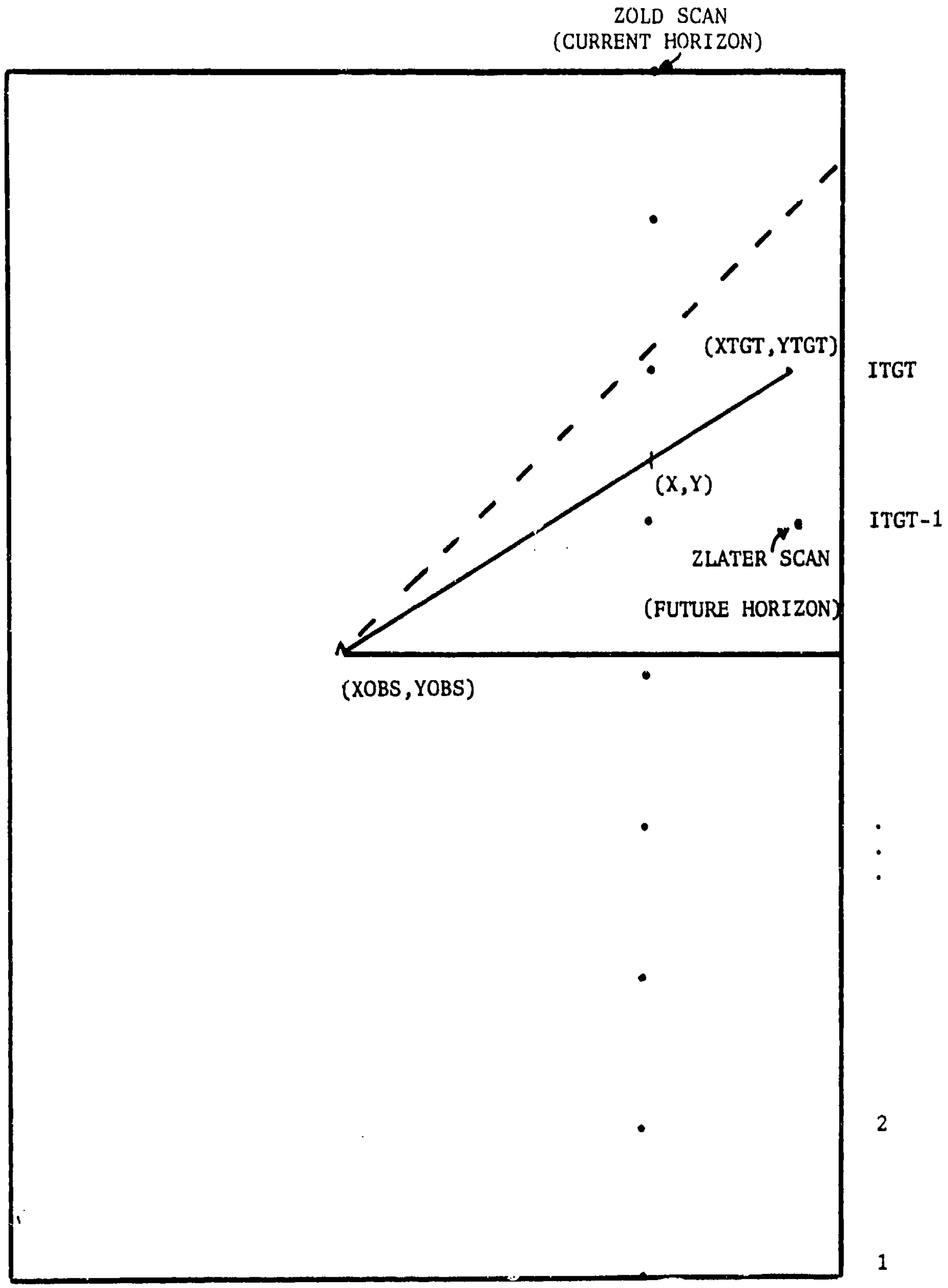


Figure B1-3

Solving for X,Y,Z when XTGT ≥ XOBS, YTGT ≥ YOBS AND (XTGT - XOBS) ≥ (YTGT - YOBS)

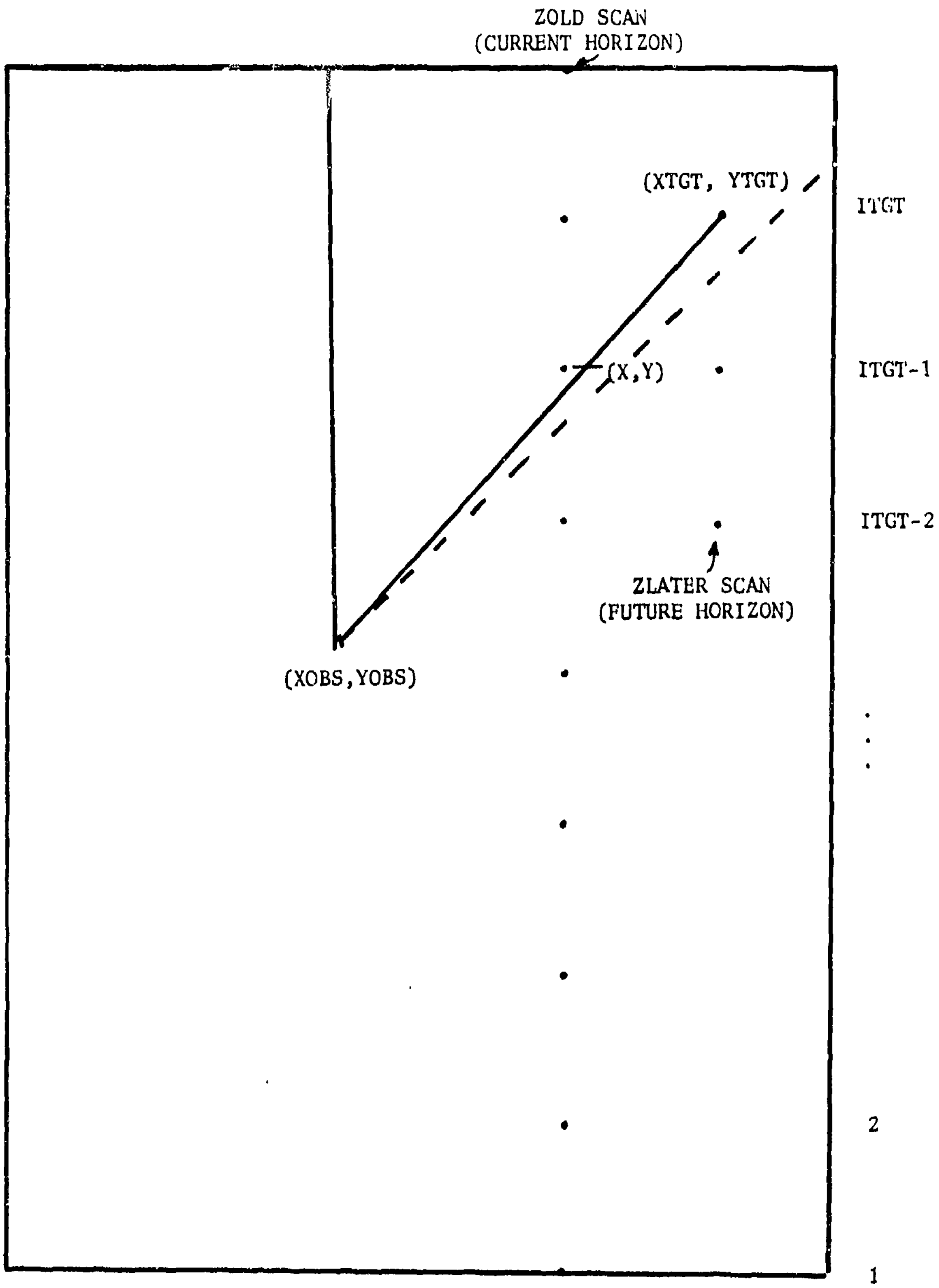


Figure B1-4

Solving for X,Y AND Z when XTGT $\geq$ XOBS, YTGT $\geq$ YOBS and

XTGT - XOBS) < (YTGT - YOBS)

Figure B1-5

Equations for X,Y,Z for the ITGTth Target Position on a Scan (Eight Possible Cases)

Sector	X	Y	Z
1U $DX \geq 0$ $DY \geq 0$ $DX < DY$	$\frac{DX(DY-GRID)}{DY}+XOBS$	YTGT-GRID	$\left(1-\frac{DX}{DY}\right)$ ZLATER(ITGT-1) $+\left(\frac{DX}{DY}\right)$ ZOLD(ITGT-1)
1D $DX \geq 0$ $DY \geq 0$ $DX > DY$	XTGT-GRID	$\frac{DY(DX-GRID)}{DX}+YOBS$	$\left(1-\frac{DY}{DX}\right)$ ZOLD(ITGT) $+\left(\frac{DY}{DX}\right)$ ZOLD(ITGT-1)
2U $DX < 0$ $DY \geq 0$ $-DX < DY$	$\frac{DX(DY-GRID)}{DY}+XOBS$	YTGT-GRID	$\left(1+\frac{DX}{DY}\right)$ ZLATER(ITGT-1) $-\left(\frac{DX}{DY}\right)$ ZOLD(ITGT-1)
2D $DX < 0$ $DY \geq 0$ $-DX \geq DY$	XTGT + GRID	$\frac{DY(DX+GRID)}{DX}+YOBS$	$\left(1+\frac{DY}{DX}\right)$ ZOLD(ITGT) $-\left(\frac{DY}{DX}\right)$ ZOLD(ITGT-1)
3U $DX < 0$ $DY < 0$ $DX < DY$	XTGT + GRID	$\frac{DY(DX+GRID)}{DX}+YOBS$	$\left(1-\frac{DY}{DX}\right)$ ZOLD (ITGT) $+\left(\frac{DY}{DX}\right)$ ZOLD(ITGT+1)
3D $DX < 0$ $DY < 0$ $DX \geq DY$	$\frac{DX(DY+GRID)}{DY}+XOBS$	YTGT + GRID	$\left(1-\frac{DX}{DY}\right)$ ZLATER(ITGT+1) $+\left(\frac{DX}{DY}\right)$ ZOLD(ITGT+1)
4U $DX \geq 0$ $DY < 0$ $DX \geq -DY$	XTGT-GRID	$\frac{DY(DX-GRID)}{DX}+YOBS$	$\left(1+\frac{DY}{DX}\right)$ ZOLD(ITGT) $-\left(\frac{DY}{DX}\right)$ ZOLD(ITGT+1)
4D $DX \geq 0$ $DY < 0$ $DX < -DY$	$\frac{DX(DY+GRID)}{DY}+XOBS$	YTGT + GRID	$\left(1+\frac{DX}{DY}\right)$ ZLATER(ITGT+1) $-\left(\frac{DX}{DY}\right)$ ZOLD(ITGT+1)

(XTGT, YTGT) = THE TARGET COORDINATES
(XOBS, YOBS) = THE OBSERVER COORDINATES
GRID = MAP GRID
DX = XTGT - XOBS
DY = YTGT - YOBS
ZOLD = ARRAY HOLDING HORIZON REPRESENTING PREVIOUS ELEVATION
ZLATER = FUTURE HORIZON VALUES CALCULATED FOR CURRENT SCAN

Appendix B2

In Appendix B1, a method was shown for calculation of x, y, and z, where (x, y) is the intersection of the observer-target line with the last running horizon line and z is the horizon elevation at the position (x, y). This Appendix describes the method for projecting the horizon elevation to the target position. It is at this time that earth curvature is introduced into the line of sight calculations.

Letting R_E = the radius of the earth

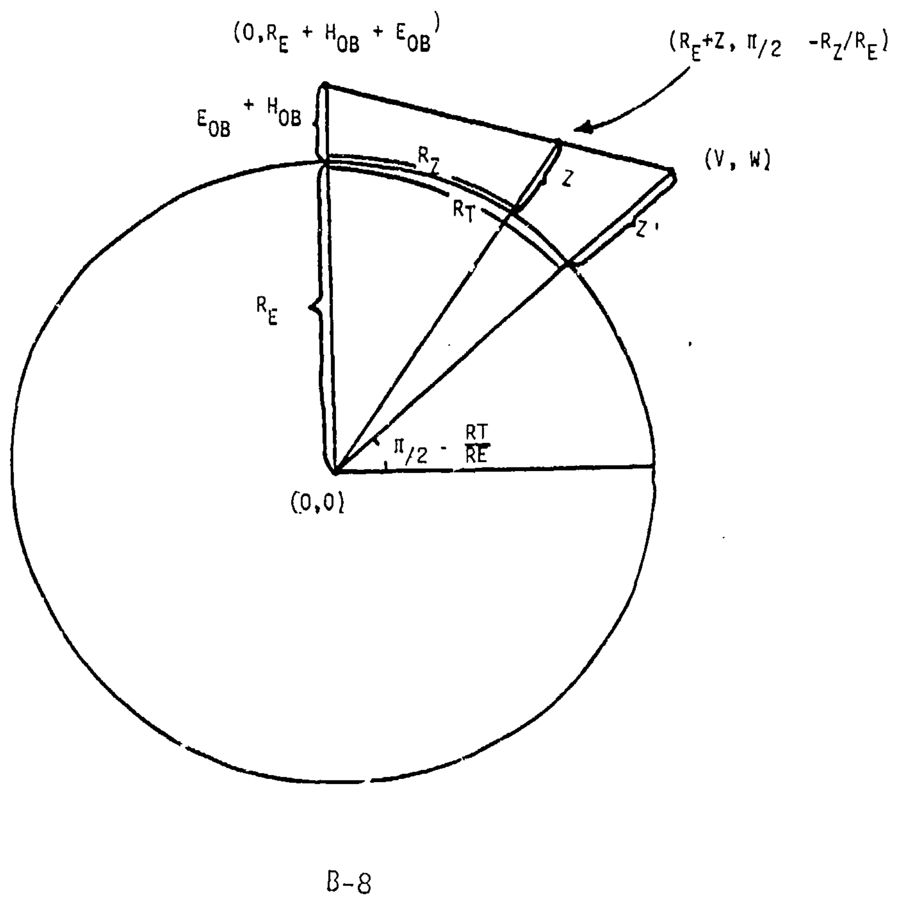
E_{OB} = the elevation of the observer

H_{OB} = the height of the observer

R_Z = the range to the horizon line from the observer

and R_T = the range to the target from the observer

the situation might be drawn as follows:



Now

$$Z' = \sqrt{V^2 + W^2} - R_E$$

so if we can determine values V and W, we can find Z'.

But (V,W) is simply the intersection of lines (0,0) - (V,W) and $(0, R_E + H_{OB} + E_{OB})$ - (V, W).

The equation for (0,0) - (V,W) is

$$Y' = AX' + B$$

where

$$B = 0$$

$$A = \tan\left(\pi/2 - \left(^{R}T/R_E\right)\right) = \cot\left(^{R}T/R_E\right)$$

And the equation for line $(0, R_E + H_{OB} + E_{OB})$ -(V, W) is

$$Y' = A'X' + B'$$

where

$$B' = R_E + H_{OB} + E_{OB}$$

and

$$A' = \frac{(R_E + Z)\cos\left(R_{Z/R_E}\right) - (R_E + H_{OB} + E_{OB})}{(R_E + Z)\sin\left(R_{Z/R_E}\right)}$$

Solving these two line equations simultaneously, then, for V and W

$$V = \frac{B'}{\cot\left(^{R}T/R_E\right) - A'}$$

$$W = \frac{\cot\left(^{R}T/R_E\right) B'}{\cot\left(^{R}T/R_E\right) - A'}$$

with A' and B' as defined above.

The next page is blank

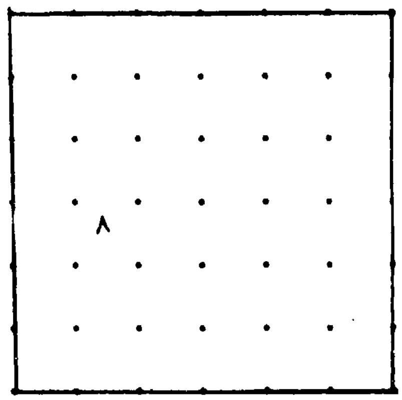
APPENDIX C

LOSMAP TIME COMPLEXITY

The next page is blank.

APPENDIX C 1

The Time Complexity for an NXN Map Using the LOSMAP Algorithm



Consider a Map with N scan lines, and N points per scan line and an observer at a point such than n scan lines are to his "left" and m points on each scan are below him.

In example above, $N = 7$

$n = 2$

$m = 3$

If a calculation has to be made each time a profile crosses either a vertical or horizontal scan line, how many calculations have to be made to compute a LOSMAP?

There are N-n scan lines to the right of the observer. The first one (the one immediately to the observer's right) requires N crossings of vertical scan lines (one for each point on the scan); the second requires 2N crossings of vertical scan lines, etc. for a total number of crossings of

$$N + 2N + 3N + \dots + (N-n) N$$
$$= N \{1 + 2 + 3 + \dots + (N-n)\}$$
$$= N \frac{(N-n)(N-n+1)}{2}$$

There are n scan lines to the left of the observer.

The total number of crossings of these is

$$N + 2N + \dots + n N$$
$$= N (1+2+\dots+n)$$
$$= N\left\{\frac{n(n+1)}{2}\right\}$$

Therefore the total number of crossings of vertical scan lines is

$$n_v = \frac{N}{2} \{(N-n)\ (N-n+1) + n(n+1)\}$$

$$n_v = \frac{N}{2} \{N^2-nN+N-nN+n^2-n+n^2+n\}$$

$$= \frac{N}{2} \{N^2+N+2n^2-2nN\}$$

$$= \frac{N}{2} \{N^2+N+2n\ (n-N)\}$$

Now $2n\ (n-N)$ is most negative when $n = N/_2$,

Since $f\ (n) = 2n^2 - 2nN$

$f'(n) = 4n - 2N$

Setting $4n - 2N = 0$, $n = {}^N/2$

So $f(n)$ has its minimum or maximum at $N/2$

But $f''(N/2) = 4>0$ so $f(n)$ is minimal when $n = N/2$.

So $n_v \geq {}^N/2\ \{N^2 + N + 2({}^N/2)\ (-{}^N/2)\}$

$\geq {}^N/2\ \{N^2/2 + N\}$

$> N^3/4 + N^2/2$, order N^3

We have only looked at vertical scan lines. There are as many again crossings of horizontal scan lines, (similar analysis using m in place of n). This essentially doubles the number of calculations, but it still remains of order N^3 for a map with minimum crossings; that is, of order $\geq N^3$.

There are always less than 2N crossings (horizontal and vertical) for each of the N^2 target positons, so the order is $\leq N^3$.

Therefore the order is N^3 for the LOSMAP algorithm.

The next page is blank

DISTRIBUTION LIST

No. of Copies	Organization
1	Commander US Army Materiel Development & Readiness Command ATTN: DRCCP DRCDE-F DRCRE-I DRCPA-S DRCQA DRCDE-R DRCDE-D DRCBSI-L DRCBSI-D 5001 Eisenhower Avenue Alexandria, VA 22333
1	Commander Harry Diamond Laboratories Lab 100 (ATTN: Saunders) 2800 Powder Mill Road Adelphi, MD 20783
2	Directorate US Army TRADOC Systems Analysis Activity ATTN: ATAA-TFC(Olson) (McCoy) White Sands Missile Range, NM 88002
1	Commander Naval Weapons Center Code 3175(ATTN: Carol Burge) China Lake, CA 93555
1	Commander & Director USAE-WES ATTN: Dr. V. LaGarde P.O. Box 631 Vicksburg, MS 39180
1	Commander Combined Arms Center & Fort Leavenworth ATTN: ATZLPTS-IS(Pfortmiller) Ft. Leavenworth, KS 66027
1	Lawrence Livermore Laboratories ATTN: Don Patterson Livermore, CA 94550
1	Commander & Director US Army Engineer Topographic Labs ATTN: James Jancaitus Ft. Belvoir, VA 22060
1	Lincoln Laboratory Massachusetts Institute of Technology P.O. Box 73 Lexington, MA 02173
1	Electromagnetic Compatability Analysis Center ATTN: S.R. Marsh, Jr. Annapolis, MD 21402
1	Commanding General US Army MICOM ATTN: DRSMI-DS(Dr. Dorsett) Redstone Arsenal, AL 35898

Aberdeen Proving Ground, MD

No. of Copies	Organization
1	Dir, HEL Bldg 520
6	Ch, TOAO Bldg 392